

The MPC Adventures: Experiences with the Generation of VLSI Design and Implementation Methodologies*

Lynn Conway

Xerox Corporation, Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304, USA

During the early '70's, Carver Mead began a pioneering series of courses in integrated circuit design at Caltech, presenting the basics of industry MOS design practice at the time. Observing some of the students' successes in later doing projects using these basics, Mead sensed that it might be possible to create new, much simpler methods of IC design than those then used in industry.

In the mid '70's, Carver Mead and Lynn Conway, and their research groups at Caltech and Xerox, began a collaboration to search for improved, simplified methods for VLSI system design. They hoped to create methods that could be very easily learned by digital system designers, but that would also allow to full architectural potential of silicon to be realized. Their research yielded important basic results during '76 and '77. In the summer of '77, they began writing the textbook *Introduction to VLSI Systems*, to document the new methods.

In the late '70's, Lynn Conway realized the need for large-scale experimentation to further generate, test, and validate the methods. Conway began using novel methods within a systematic, rapidly expanding set of interactions with many universities throughout the United States. Students at these schools took courses using the evolving textbook, and then did design projects as part of those courses. The projects were implemented, and the resulting feedback was used to extend, refine, and debug the text, the courses, the university design environments, and the new design methods.

As a result of the research methodology used, and the very large scale of the interactions with the university community (via computer-communications networks), the Mead-Conway design methods evolved unusually rapidly, going from concept to integration within industry in just a few years.

This talk tells the story of these events, focussing on the research methods used to generate, validate, and culturally integrate the Mead-Conway design methods.

Keywords: VLSI design, Mead-Conway design method.

Copyright © 1981, Lynn Conway. All Rights Reserved.

* This paper is a slightly edited transcription from an Invited Lecture at the Second Caltech Conference on Very Large Scale Integration (January 19, 1981).

1. Introduction

This paper is about "The MPC Adventures", namely the multi-university, MultiProject Chip escapades of the past few years. I'll describe these adventures, and the new VLSI implementation system that made possible the economical, fast-turnaround implementation of VLSI design projects on such a large scale. I'll also describe the experiences I've had with the processes involved in generating new cultural forms such as the "Mead-Conway" VLSI design and implementation methodologies. One of my objectives here is to help you visualize the role that the "MPC Adventures" played in the generation of the methodologies.

I am particularly interested in developing effective research methodologies in the sciences of the artificial, especially in areas such as engineering design. The sort of question that really interest me is: How can we best organize to create, validate, and culturally integrate new design methods in new technologies? What are the research dynamics involved? Consider the following:

When new design methods are introduced in any technology, especially in a new technology, a large-scale exploratory application of the methods by many designers is necessary in order to test and validate the methods. A lot of effort must be expended by a lot of people, struggling to create many different systems, in order to debug the primitives and composition rules of the methodology and their interaction with the underlying technology. A similar effort must also be expended to generate enough design examples to evaluate the architectural possibilities of the design methods and the technology. That is the first point: *A lot of exploratory usage is necessary to debug and evaluate new design methods.* The more explorers

that are involved in this process, and the better they are able to communicate, the faster the process runs to any given degree of completion.

Suppose some new design methods have been used and fairly well debugged by a community of exploratory designers, and have proven very useful. Now consider the following question: How can you take methods that are new, methods that are not in common use and therefore perhaps considered *unsound methods*, and turn them into *sound methods*? In other words, how can you cause the *cultural integration* of the new methods, so that the average designer feels comfortable using the methods, considers such usage to be part of their normal duties, and works hard to correctly use the methods? Such cultural integration requires a major shift in technical viewpoints by many, many individual designers. Changes in design practices usually require changes in the social organization in which the designer functions. These are difficult obstacles to overcome. We see that numbers are important again, leading us to the second point: *A lot of usage is necessary to enable sufficient individual viewpoint shifts and social organization shifts to occur to effect the cultural integration of the methods.* The more designers involved in using the new methods, and the better they are able to communicate with each other, the faster the process of cultural integration runs.

When methods are new and are still considered unsound, it is usually impossible in traditional environments to recruit and organize the large numbers of participants required for rapid, thorough exploration and for cultural integration. Therefore, new design methods normally evolve via rather ad hoc, undirected processes of cultural diffusion through dispersed, loosely connected groups of practitioners, over relatively long periods of time. (Think, for example of the effect of the vacuum-tube-to-transistor technology transition on the design practices of the electronic design community, or of the effect of the discrete-transistor-to-TTL technology transition). When the underlying technology changes in some important way, new design methods exploiting the change compete for market share of designer mind-time, in an ad hoc process of diffusion. Bits

and pieces of design lore, design examples, design artifacts, and news of successful market applications, move through the interactions of individual designers, and through the trade and professional journals, conferences, and mass media. When a new design methodology has become widely integrated into practice in industry, we finally see textbooks published and university courses introduced on the subject.

I believe we can discover powerful alternatives to that logn, ad hoc, undirected process. Much of this talk concerns the application of methods of experimental computer science to the particular case of the rapid directed creation, validation, and cultural integration of the new VLSI design and VLSI implementation methods within a large computer-communication network community.

First I will sketch the evolution of the new VLSI design methods, the new VLSI design courses, and the role that implementation played in validating the concepts as they evolved. Next I'll bring you up to date on the present status of the methods, the courses, and the implementation systems. Finally, I'll sketch the methods that were used to direct this evolutionary process. We'll reflect a bit on those methods, and look ahead to other areas where such methods might be applied.

2. Evolution of the VLSI Design Courses: Role of the MPC Adventures

In the early 1970's, Carver Mead began offering a pioneering series of courses in integrated circuit design at Caltech. The students in these courses in MOS circuit design were presented the basics of industrial design practice at the time. Some of these students went on to do actual design projects, and Carver found that even those without backgrounds in device physics were able to complete rather ambitious projects after learning these basics. These experiences suggested that it might be feasible to create new and even simpler methods of integrated system design.

In the mid 1970's, a collaboration was formed between my group at Xerox PARC and a group led by Carver Mead at Caltech, to search for improved methods for VLSI design. We undertook an effort

to create, document, and debug a simple, complete, consistent method for digital system design in nMOS. We hoped to develop and document a method that could be quickly learned and applied by digital system designers, folks skilled in the problem domain (digital system architecture and design) but having limited backgrounds in the solution domain (circuit design and device physics). We hoped to generate a method that would enable the system designer to really exploit the architectural possibilities of planar silicon technology without giving up the order of magnitude or more in area-time-energy performance sacrificed when using the intermediate representation of logic gates as in, for example, traditional polycell or gate-array techniques.

Our collaborative research on design methodology yielded important basic results during '76 and '77. We formulate some very simple rules for composing FET switches to do logic and make registers, so that system designers could easily visualize the mapping of synchronous digital systems into nMOS. We formulated a simple set of concepts for estimating system performance. We created a number of design examples that applied and illustrated the methods.

2.1. The Mead-Conway Text

Now, what could we do with this knowledge? Write papers? Just design chips? I was very aware of the difficulty of bringing forth a new *system of knowledge* by just publishing bits and pieces of it in among traditional work.

I suggested the idea of writing a book, actually of *evolving a book*, in order to generate and integrate the methods, and in August 1977 Carver and I began work on the Mead-Conway text. We hoped to document a complete, but simple, system of design knowledge in the text, along with detailed design examples. We quickly wrote preliminary draft of the first three chapters of this text, making use of the Alto personal computers, the network, and the electronic printing systems at PARC. In parallel with this, Carver stimulated work on an important design example here at Caltech, the work on the "OM2". Dave Johannsen carefully applied the new design methods as they were being

documented, refined and simplified, to the creation of this major design example.

We then decided to experimentally debug the first three chapters of material by interjecting them into some university MOS design courses. An initial draft of the first three chapters [1a] was used by Carlo Sequin at U.C. Berkeley, and by Carver Mead at Caltech in the fall of '77. During the fall and winter of '77-'78, Dave Johannsen finished and documented the new OM2 design. The OM2 provided very detailed design examples that were incorporated into a draft of the first five chapters [1b] of the text. We distributed that draft in February '78 into spring semester courses by Bob Sproull at CMU, and by Fred Rosenberger at Washington University, St. Louis.

We were able to debug and improve the material in these early drafts by getting immediate feedback from the '77-'78 courses. We depended heavily on use of the ARPAnet for electronic message communications. Our work rapidly gained momentum. A number of people joined to collaborate with us during the spring of '78: Bob Sproull at CMU and Dick Lyon at PARC created the CIF 2.0 specification; Chuck Seitz prepared the draft of Chapter 7 on self-times systems; H.T. Kung and several others contributed important material for Chapter 8 on Concurrent Processing. By the summer of '78 we completed a draft of the manuscript of the entire textbook [1c].

2.2. The MIT'78 VLSI Design Course

During the summer of 1978, I prepared to visit M.I.T. to introduce the first VLSI system design course there. This was to be a major test of the full set of new methods and of a new intensive project-oriented form of course. I also hoped to thoroughly debug the text prior to publication. I wondered: How could I really test the methods and test the course contents? The answer was to spend only half of the course on lectures on design methods; then in the second half, have the students do design projects. I'd then try to rapidly implement the projects and see if any of them worked (and if not, find out what the bugs were). That way I could discover bugs, or missing knowledge, or missing constraints in the design methods or in the course contents.

I prepared a detailed outline for such a course, and printed up a bunch of the drafts of the text. Bob Hon and Carlo Sequin organized the preparation of a "Guide to LSI Implementation" [2] that contained lots of practical information related to doing projects, including a simple library of cells for I/O pads, PLA's, etc. I then travelled to M.I.T., and began the course. It was a very exciting experience, and went very well. We spent seven weeks on design lectures, and then an intensive seven weeks on the projects. Shortly into the project phase it became clear that things were working out very well, and that some amazing projects would result from the course.

While the students were finishing their design projects, I cast about for a way to get them implemented. I wanted to actually get chips made so we could see if the projects worked as intended. But more than that, I wanted to see if the whole course and the whole method worked, and if so, to have demonstrable evidence that it had. So I wanted to take the completed layout descriptions and very quickly turn them into chips, i.e. implement the designs (We use the term "VLSI implementation" for the overall process of merging the designs into a starting frame, converting the data into patterning format, making masks, processing wafers, dicing the wafers into chips, and mounting and wire-bonding the chips into packages).

We were fortunate to be able to make arrangements for fast implementation of those student projects following the MIT course. I transmitted the design files over the ARPAnet from M.I.T. on the east coast to some folks in my group at PARC on the west coast. The layouts of all the student projects were merged together into one giant multi-project chip layout, a trick developed here at Caltech, so as to share the overhead of mask-making and wafer fab over all of the designs. The project set was then hustled rapidly through the prearranged mask and fab services. Maskmaking was done by Micro-Mask, Inc., using their new electron-beam maskmaking system, and wafer fabrication was done by Pat Castro's Integrated Circuit Processing Lab (ICPL) at HP Research, in Palo Alto. We were able to get the chips back to the students about six weeks after the end of the course. A number of the M.I.T. '78 projects work-

ed, and we were able to uncover what had gone wrong in the design of several of those that didn't.

The M.I.T. course led to a very exciting group of projects, some of which have been described in later publications. The project by Jim Cherry, a transformational memory system for mirroring and rotating bit map image data, is particularly interesting, and was one of those that worked completely correctly. Jim's project is described in detail in the second edition of the Hon and Sequin Guidebook (see Ref. [5]). Another interesting project is the prototype LISP microprocessor designed by Guy Steele, that was later described in an M.I.T. AI Lab report [3].

As a result of this course and the project experiences, we uncovered a few more bugs in the design methods, found constraints that were not specified, topics that were not mentioned in the text, that sort of thing. You can see that the project implementation did far more than test student projects. It also tested the design methods, the text, and the course.

During the spring of '79 we began preparing the final manuscript of the Mead-Conway text for publication by Addison-Wesley the following fall [4]. Hon and Sequin began preparing a major revision of the Implementation Guide [5] that would contain important things like a CIF primer, new, improved library cells, and so forth. I began preparing an "Instructor's Guide", based on the experiences and information from the M.I.T. '78 VLSI design course [6], containing a detailed course outline, a complete set of lecture notes, and homework assignments from that course. These materials would help transport the course to other environments.

2.3. The MPC Adventures: MPC79 and MPC580

I'll now describe the events surrounding the multi-project chip network adventures of the fall of 1979 and spring of 1980. I remember thinking: "Well, ok, we've developed a text, and also a course curriculum that seems transportable. The question now is, can the course be transported to many new environments? Can it be transported without one of the principals running the course?" In reflecting on the early work on the text by communicating

with our collaborators via the ARPAnet, and by thinking about which schools might be interested in offering courses, I got an idea: If we could find ways of starting project-oriented courses at several additional schools, and if we could also provide VLSI implementation for all the resulting student projects, we could conduct a really large test of our methods. The course might be successful in some schools, and not in others, and we could certainly learn a lot from those experiences. I began to ponder the many ways we would use the network to conduct such an adventure.

We began to train instructors from a number of universities in the methods of teaching VLSI design. Doug Fairbairn and Dick Lyon ran an intensive short course for PARC researchers during the spring of '79, and a videotape [7] was made of that entire course. During the summer of '79, we began using those tapes as the basis for short, intensive "instructor's courses" at PARC for university faculty members. Carver Mead and Ted Kehl also ran an instructor's course at the University of Washington, with the help of the PARC tapes, in the summer of '79. All "graduates" of the courses received copies of the Instructor's Guide, to use as a script at their schools.

By early fall of '79, quite a few instructors were ready to offer courses. We at PARC gathered up our nerve, and then announced to this group of universities: "If you run courses, we will figure out some way so that the end of your course, on a specified date, we will take in any designs that you transmit to use over the ARPAnet; we will implement those projects, and send back wirebonded, packaged chips for all of your projects within a month of the end of your course!" This multi-university, multiproject chip implementation effort came to be known as "MPC79".

About a dozen universities joined to participate in MPC79. At this large university community became involved, the project took on the characteristics of a great "network adventure", with many people simultaneously doing large projects to test our new ideas. Through the implementation effort, students hoped to validate their design projects, instructors would be able to validate their offering of the course, and we would be able to further validate and test the design methodology and

the new implementation methods in development at PARC.

We coordinated the MPC79 events by broadcasting a series of detailed "informational messages" out over the network to the project lab coordinators at each school. MSG #1 announced the service and the schedule; MSG #2 distributed the basic library cells, including I/O pads and PLA cells; MSG #3 described the "User's Guide" for interactions with the system; MSG #4 contained information about the use of CIF2.0; MSG #5 provided last-minute information just prior to the design deadline; MSG #6 was sent just after the implementation was completed, and contained news about the results of the entire effort. Fig. 1 flowcharts the overall activity.

During this period, Allan Bell pioneered the architecture and teamed up with Martin Newell to develop a "VLSI Implementation System", which is something like a time-sharing operating system, or information management system, for providing remote access to mask and fab services. This system manages all user interactions, manages the data base of design files, handles the logistics, the scheduling enabling users all around the country to interact by electronic messages with (what they perceive to be) an automatic system that implements their projects.

Fig. 2 shows a simple block diagram of the basic modules of the system. It contains a user message handler and an associated design file processing subsystem; these provide a means for interacting with users to receive requests for service, transmit status and error messages, and build the design-file data base. It also contains a die-layout planning and design-file merging subsystem used to pack all of the participants designs together into a mask specification following the design deadline time. Finally it contains a CIF to MEBES (electron beam maskmaking) format-conversion subsystem to prepare the data files for hand off to the foundry.

Following is a photo (Fig. 3) of Alan Bell operating the implementation system at PARC during the very final stages of project emerging following the MPC79 design deadline. He's taken almost all of the designs, as identified in a display menu listing the project ID's, and packed them into the 12 die-types of the project set.

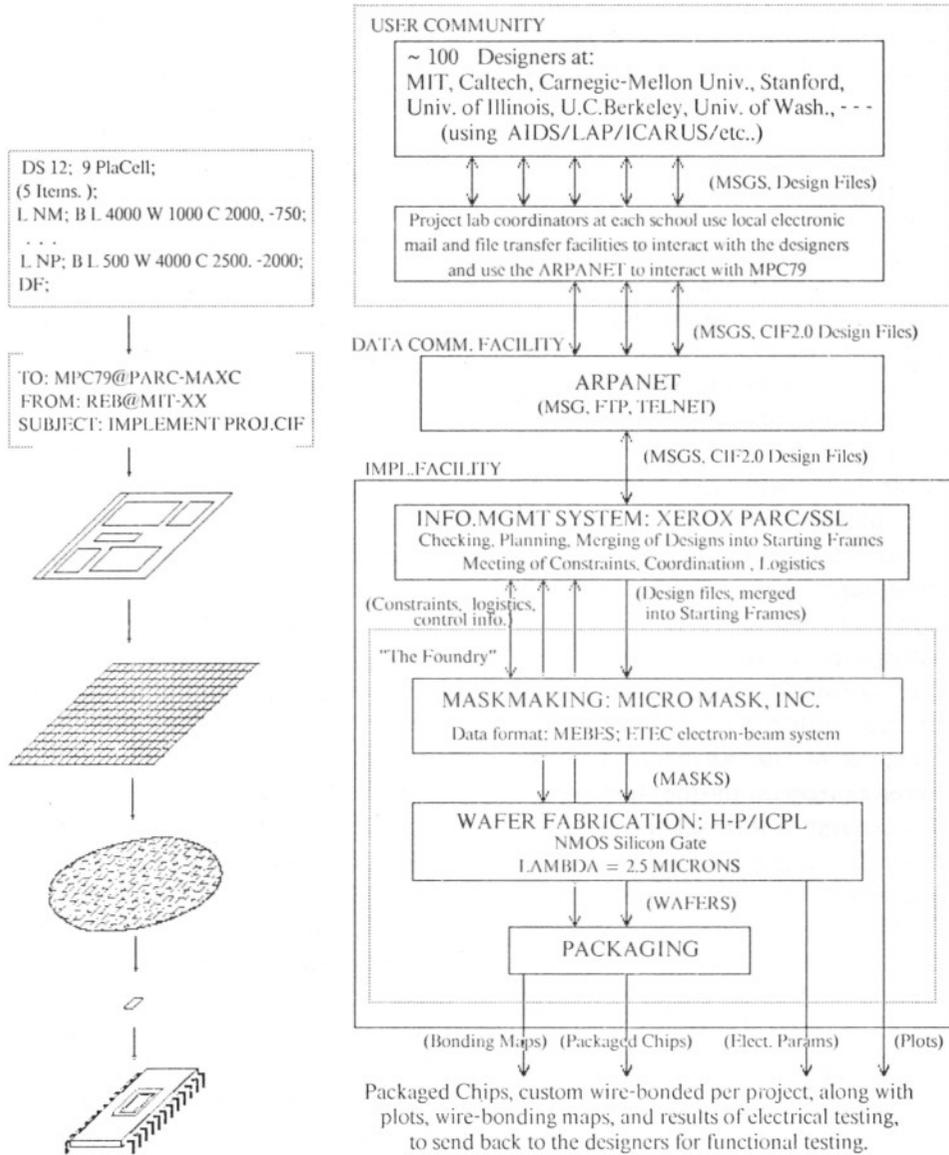


Fig. 1. MPC79 Flowchart.

For MPC79, the implementation system produced MEBES mask specifications containing 82 projects from 124 participating designers merged into 12 die-types that were distributed over two mask sets. Thus there was a tremendous sharing of the overhead involved in the maskmaking and wafer fab. For MPC79 the masks were again made by Micro-Mask, Inc., and wafer fabrication was again done by HP-ICPL. Several chips of each

project types were custom wire-bonded and prepared for shipment back to the designer, along with "implementation documentation" [8] containing pinout information for the projects, electrical parameter measurements for the wafer lots, etc. Fig. 4 provides a visualization of the many projects conveyed through one of the MPC79 wafer types, and of the corresponding hierarchy of information associated with the project set.

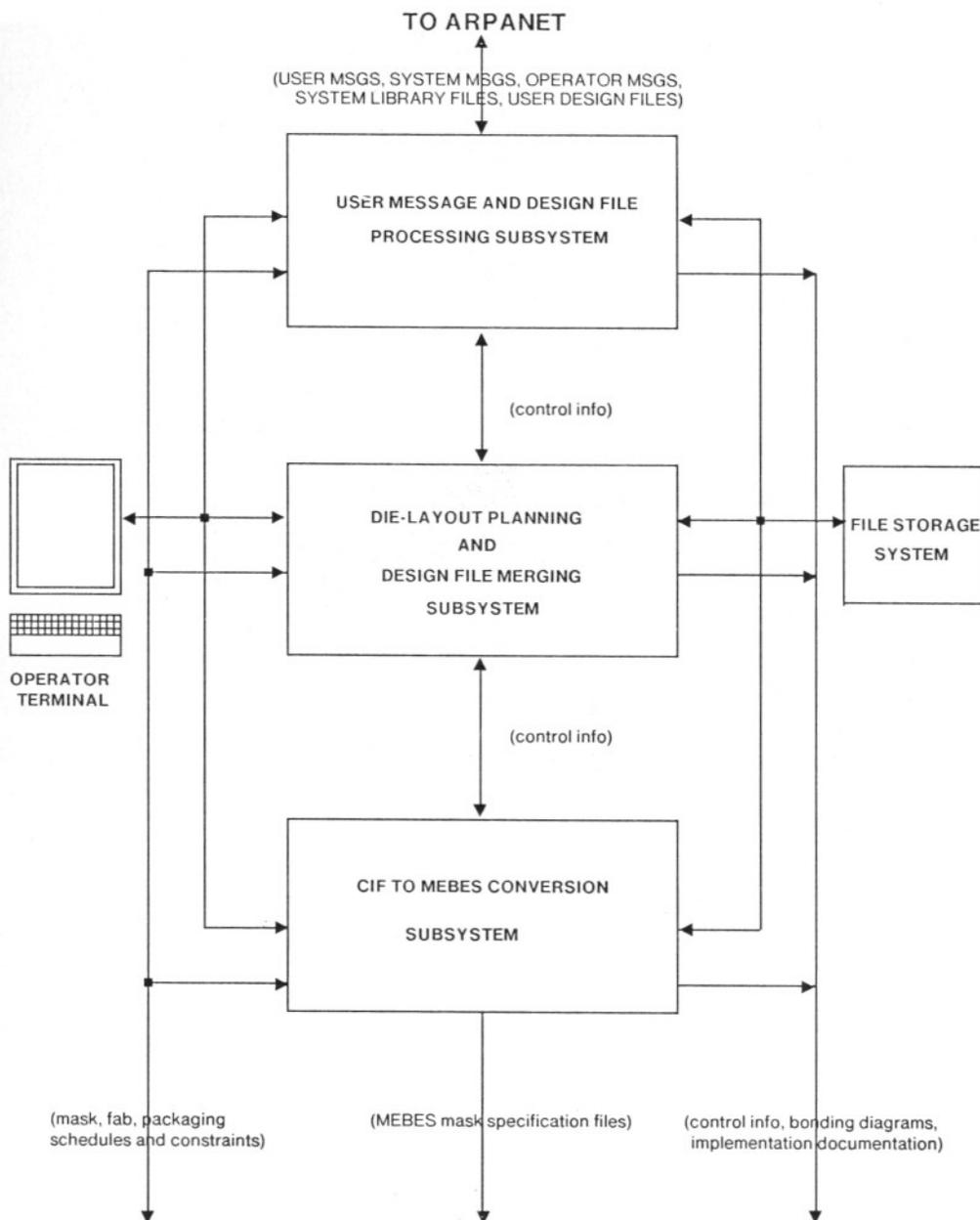


Fig. 2. Block Diagram of the VLSI Implementation System.

Just 29 days after the design deadline time at the end of the courses, packaged custom wire-bonded chips were shipped back to all the MPC79 designers. Many of these worked as planned, and the overall activity was a great success. Examples of the many interesting MPC79 projects can be seen in the photo of one of the multiproject chips

produced by students and faculty researchers at Stanford University (Fig. 5). Among these is the first prototype of the "Geometry Engine", a high-performance computer graphics image-generation system, designed by Jim Clark. That project has since evolved into a very interesting architectural exploration and development project [9].

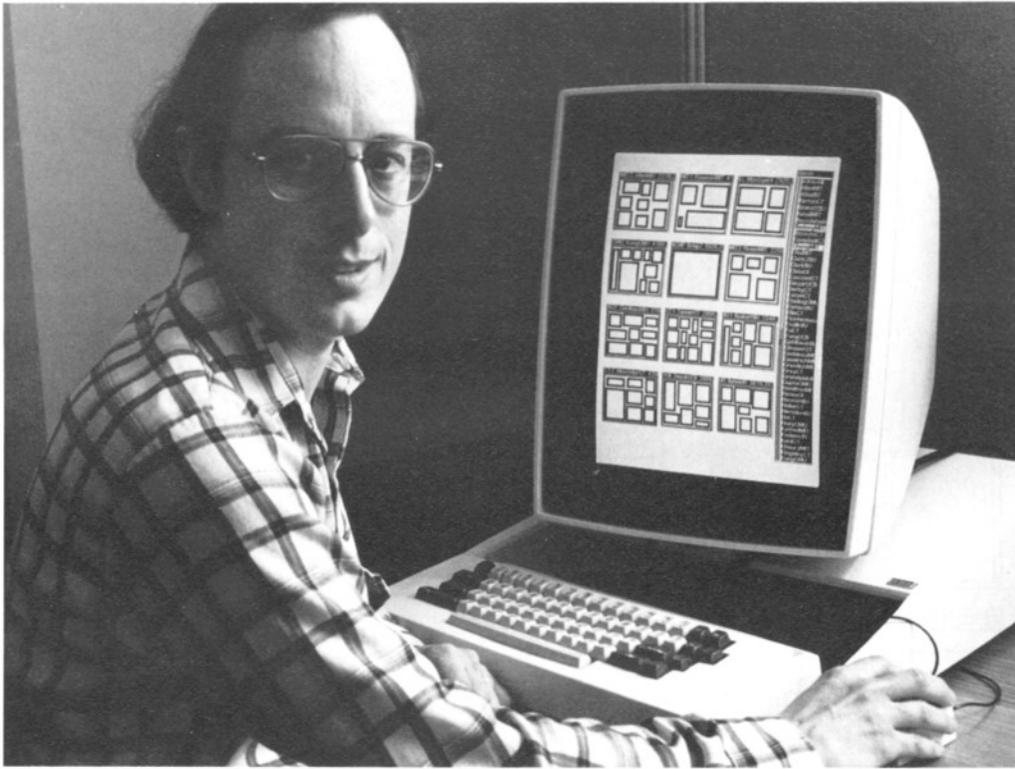


Fig. 3. Alan Bell using the Implementation System to merge the MPC79 Projects.

Another project that turned up in MPC79 was a LISP microprocessor [10] designed by Holloway, Sussman, and Steele at MIT and Bell at PARC. This "Scheme-79" chip is a further step in the evolution of LISP microprocessor architectures by the M.I.T. AI-Lab group. Their work is based on the prototype LISP microprocessor [3] Guy Steele designed for the 1978 MIT course.

The results of this design methodology experimentation and demonstration were very exciting, and convinced us of the overall merits of the design methods, the courses, and the implementation infrastructure. We first reported on the results at the M.I.T. VLSI conference in January 1980 [11,12].

At PARC we then began the transfer of the implementation system technology to an internal operational group; the transfer was completed during the spring of 1980. That operational group now has the responsibility of providing VLSI implementation service within Xerox. They ran the implementation system for a very large group of

schools in the spring of 1980, in order to provide themselves with a full-scale test the overall operation of the system, and to confirm the success of the technology transfer. That effort, known as "MPC580" [13], had about twice as many participants as did MPC79. Over 250 designers were involved! They produced so many projects, including a number of full-die sized projects, that 5 mask sets were required. Although MPC580 involved a lot of maskmaking and wafer fabrication, the project set was turned around from design-cutoff to packaged chips in about six weeks.

Some really interesting projects were created by the MPC580 designers. An example is the RSA encryption chip [14] designed by Ron Rivest at MIT. Ron is a computer science theoretician and faculty member at M.I.T., had taken the VLSI design course the previous fall, and had done a small project for MPC79. He and several other M.I.T. people then created the prototype RSA encryption chip architecture and design during the spring of 1980, in time for the MPC580 cutoff.

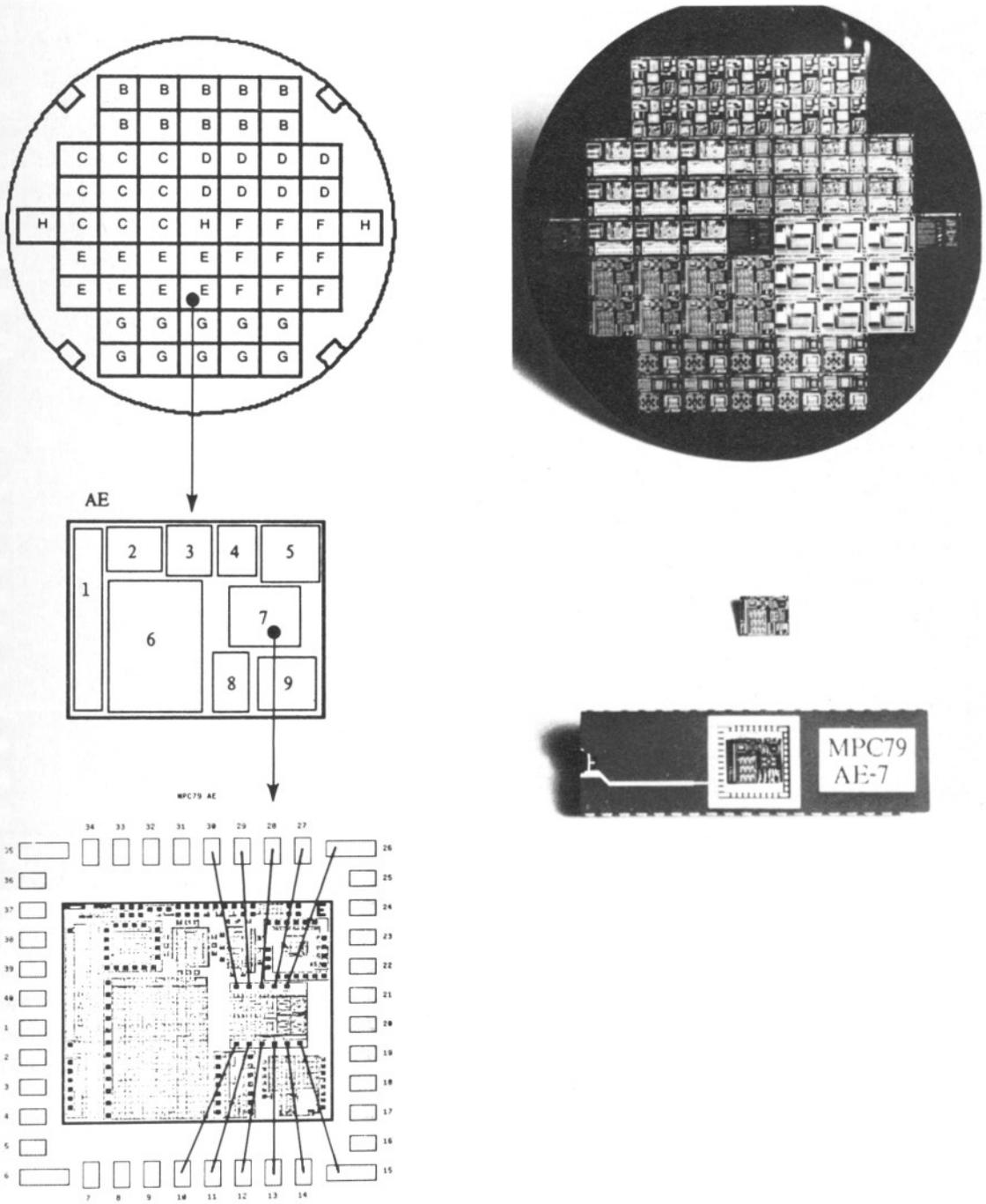


Fig. 4. At right: Photo of MPC79 type-A wafer, type-AE die, type AE-7 chip. At left: Corresponding hierarchy of informational material.

I think you can now begin to see the role the provision of implementation plays in stimulating ar-

chitectural exploration, the offering of design courses, and the creation of design environments.

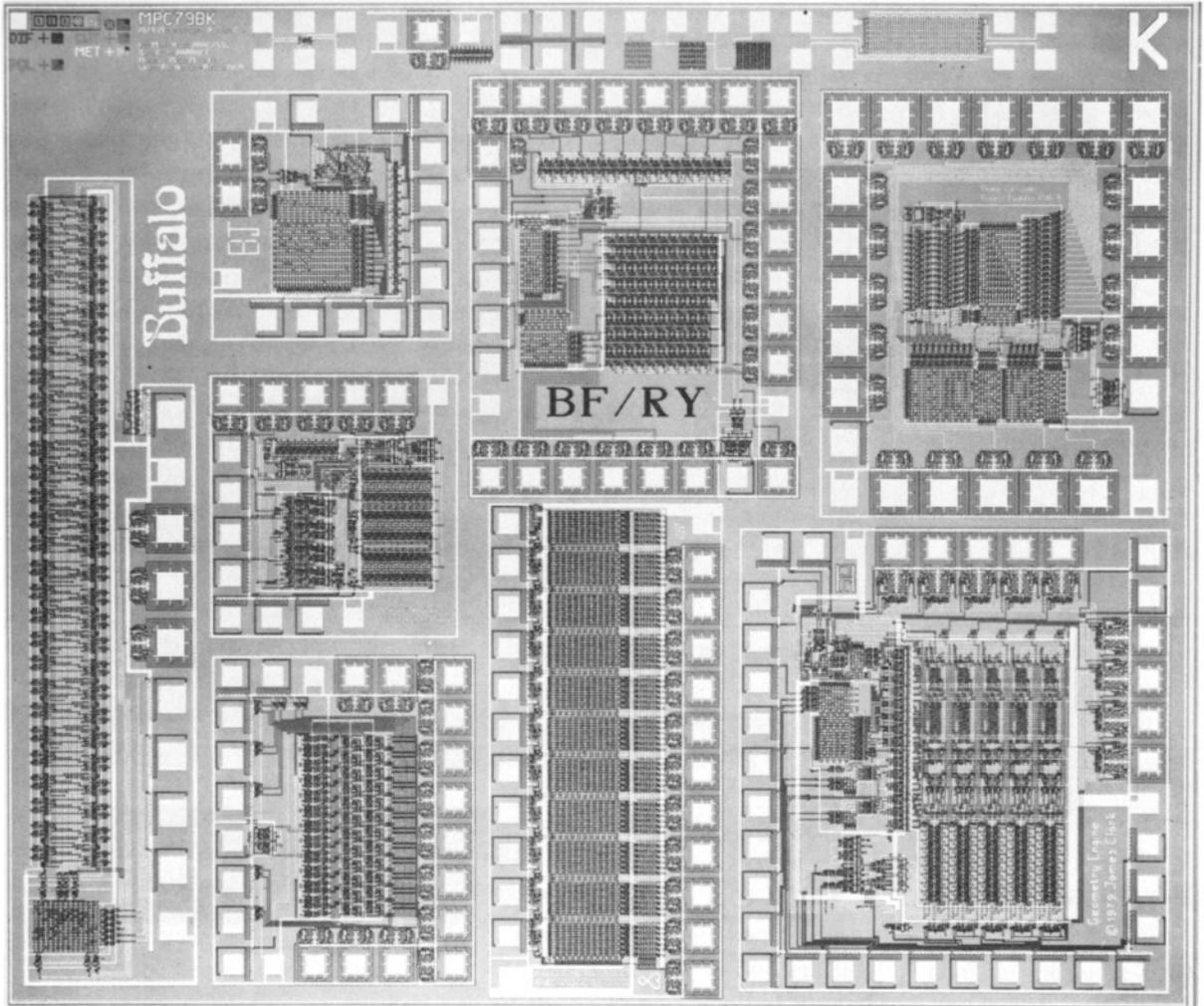


Fig. 5. Photo of MPC79 Die-Type BK (containing Projects from Stanford University).

3. Present Status of the VLSI Design Courses and the VLSI Implementation Systems

The design methodology introduced in the Mead-Conway text has now become well integrated into the university computer science culture and educational curriculum. During the '79-'80 school year, courses were offered at about 12 universities. By the '80-'81 school year, courses were being offered at more than 80 universities.

In addition, a number of industrial firms have begun to offer internal, intensive courses on the design methodology. For example, courses are being offered at several locations within Hewlett-

Packard, under the leadership of Merrill Brooksby, Manager of Corporate Design Aids at HP. The HP courses are project oriented, and provide students with fast-turnaround project implementation. Brooksby believes that in addition to directly improving the skills of HP designers, the course plays an important role by providing a common internal base of design knowledge through which designers can communicate about work in other technologies (the "common culture effect"). Similar courses are being offered at DEC, in an effort led by Lee Williams. Many other industrial firms have begun using an excellent videotype VLSI system design course produced

recently by VLSI Technology, Inc. (VTI) [15].

Design aid concepts and software are evolving rapidly in the university VLSI research community. During the work on MPC79, we began to see very interesting new types of analysis aids originating at MIT. I'm thinking of the work of Clark Baker, Chris Terman, and Randy Bryant who began creating circuit extractors, static checkers, and switch simulators of a sort appropriate for our design methods [16, 17]. They began to provide access to such analysis aids over the network, aids that could be easily and efficiently used to partially validate projects prior to implementation. These tools were used to debug some projects prior to submission to MPC79 (for example, the Scheme-79 chip). Some of these tools are now in routine use at a number of other universities. I believe we'll soon see analysis aids embodying these new concepts placed into widespread use in industry.

A VLSI implementation system has been put into use by Xerox Corporate Research to support exploratory VLSI system architecture and design within Xerox Corporation. Another implementation system is being operated by USC/ISI for the Defence Advance Research Projects Agency's (DARPA) VLSI research community, a community consisting of several large research universities (including M.I.T., CMU, Stanford, U.C. Berkeley, Caltech, etc.) and a number of Defense Department research contractors.

The initial system architecture of the system used for MPC79, and the operational experiences during MPC79, provided the knowledge on which the new Xerox and ISI systems were based. One of the major improvements contained in both these newer systems is the fully-automated handling of user electronic message interactions and management of the design file data base. During MPC79, Alan Bell interacted with the designers with some machine assistance in message handling (using a menu-based graphical interface that made message-processing and file management interactions easy and fast), but in fact he did actually look at all user messages. When we ran MPC79, we couldn't predict the bounds on the information that would have to be conveyed between designer and system. The generation of that knowledge was

an important result of MPC79, making it possible to automate the message handling and data base management in later systems. Our knowledge about the implementation system to foundry interface was also considerably expanded and refined during these experiences [18].

As I think back over the origins of the VLSI implementation system, it's clear that we didn't initially set out to create such a system. It was really a serendipitous result. We were extremely motivated and driven to provide VLSI implementation to a large university community. I thought that it just might be possible to do that. I realized that pulling off VLSI implementation on such a vast scale would generate and propagate a lot of artifacts, and thus announce the presence of the new design culture, and help to culturally integrate our methods. So, we began working very hard at PARC to create ideas to bring down the cost per project and the overall turnaround time, and to scale up capabilities for handling as many designers as possible.

Somewhere along the line I began to use the metaphor that "we're creating something for mask and fab that was like the time-shared operating system was for computing systems". Our idea was to create a system that provided remote-entry, time and cost-sharing access to expensive capital equipment, and that also managed the logistics of providing such access to a large user community.

At that time, and even now in most integrated circuit design environments, the maskmaking and wafer fabrication required to implement prototypes for a design project cost about \$15,000 to \$20,000, and with some luck take only three or four months getting through the various queues. (Designers using internal company facilities may not see those costs, but I guarantee they're there; on the other hand, all IC designers are familiar with those long turnaround times). With that as background, we were really amazed when we added up the costs in dollars and time to implement the projects in MPC79. By using the implementation system to provide shared access for a large community of users to what amounts of a "fast-turnaround silicon foundry" for rapid maskmaking and wafer fabrication, we achieved a cost per project on the order of a few hundred dollars,

Table 1
Computing and Design Environments for 1980-81 VLSI Design Courses at Universities that participated in MPC79/MPC580 (Reprinted with permission of *LAMBDA, The Magazine of VLSI Design* [19]).

UNIVERSITY:	MIT	Caltech	Stanford	CMU	U.C.B.	U. of Col. (C.S.)	U. of Illinois	U. of Wash.	U. of Rochester	UCLA	Wash. U. (St. L.)	U.S.C.
COURSE INFORMATION												
Instructor(s)	J. Allen, L. Glasser	C. Mead, C. Sietz	J. Newkirk, R. Mathews	R. Sproull	R. Newton C. Sequin	J. Murray	J. Abraham E. Davidson	T. Kehl	E. Kinnen G. Kedem	V. Tyree	F. Rosenberger	J. Nelson
Course #	6.371	CS181, CS182	EE271	15-846	CS248	EE594	EE325	CS590D	492	M258A, B, C	EE463	EE599
Sem. or Qtr.	F, Sp	F-W-Sp	F, Sp	Sp	F	F	F, Sp	F, W, Sp	F	F-W-Sp	Sp	F
#stud./Class	35	40	60	30	50	20	20	15	25	20	25	30
COMPUTING ENVIRONMENT												
CPU	DEC-20	DEC-20, VAX	DEC-VAX	DEC-VAX	DEC-VAX	DEC-20	HP1000	DEC-20, VAX	ALTO, VAX	DEC-VAX	DEC-20	DEC-KL10
Op. Sys.	TOPS-20	TOPS-20, UNIX	UNIX	UNIX	UNIX	TOPS-20	RTE IV	TOPS-20, VMS	ALTO, UNIX	UNIX	TOPS-20	TOPS-10
Prog. Lang.	LISP, APL, CLU	Simula, C	C	C	C	Simula, Pascal	Pascal	FORTRAN	C, Pascal	C, Pascal	Simula, FORTRAN	Pascal
DESIGN AID ENVIRONMENT												
Synthesis aids	PLAG, MI	MG, MI	PLAG	PLAG	PLAG, SGC	MG, MI	---	PLAG, MI	---	---	PLAG	---
Description aids	SLL	SLL	SLL	SLL, IGL	SLL, IGS	SLL	IGL	SLL	IGL	SLL	SLL	SLL
Analysis aids	CX, SS, DRC, CS	CX, SS, CS	CX, DRC, SS	CX, SS, DRC	CX, SS, CS	CS	CS	DRC	CX, SS, DRC	CS	CS	CS
Viewing aids	CPP, BRP	CPP, CRP, CD	BRP	CPP, BRP, CD, BD	BRP, BD	CPP	CPP, BD	CPP	CPP, BRP, CD, BD	CPP	CPP, CD	CPP
Testing aids	MTE	MTE	MTE	---	---	---	MTE	---	MTE	MTE	---	---
PROJECT EXPERIENCE												
(# projects, # designers)												
MPC79	15, 27	24, 28	19, 35	5, 5	4, 4	1, 1	5, 8	1, 3	5, 9	---	---	---
MPC580	11, 13	21, 22	32, 59	12, 17	8, 12	12, 21	8, 13	15, 15	3, 3	9, 9	9, 11	10, 15

SUMMARY OF DESIGN-AID CODES:

BD: B/W Display; BRP: B/W Raster Plotter; CD: Color Display; CPP: Color Pen Plotter; CRP: Color Raster Plotter; CS: Circuit Simulator; CX: Circuit eXtractor; DRC: layout Design Rule Checker; IGL:

Interactive Graphic Layout; IGS: Interactive Graphic Sticks; MG: Module Generator; MI: Module Interconnector; MTE: Minimal Test Environment; PLAG: PLA generator; SGC: Sticks-to-layout Generator/Compressor; SLL: Symbolic Layout Language; SS: Switch Simulators; SSL: Symbolic Sticks Language.

and a total turnaround time of only 29 days! (And remember, we weren't using internal mask and fab facilities at PARC, but were instead going to outside foundry services.)

Thus we had demonstrated that the time and cost to implement a prototype VLSI designs were as low as they would be using TTL for an equivalent size designs. However, once you've successfully prototyped a design in VLSI, you can take tremendous advantage of the low replication costs and high-performance of VLSI when competing against similar systems implemented in TTL. Therefore, I believe that in addition to the many business opportunities in VLSI design aids and chip designs, there must also be a substantial business opportunities in the area of VLSI implementation systems and services, foundry service brokerage, and foundry services.

Those of you who are interested in learning more about the present courses and design aid environments in the universities might read my recent column [19] in Lambda Magazine. Table 1 (from that article) tabulates the courses, the computing and design-aid environments (as of summer 1980), and the project experience at the key group of 12 universities that collaborated with us at PARC during MPC79 and MPC580. You can see some interesting patterns of diffusion and convergence in this table. You can see how new types of analysis aids are being used at most schools to qualify projects for implementation, and how rapidly those new concepts have swept through this university community, most of whom are on the ARPAnet.

4. Sketch of and Reflections on the Research Methods Used

How was all of this done? Let's reflect on these events, focussing on the research methods used to direct and help all of these different things jointly evolve. You'll notice a common idea running through all of these events: Fast-turnaround implementation provides a means for testing concepts and systems at many levels. It isn't just used for testing the project chips. It also tests the design environments, the courses and instructional methods, the text materials, and the design methods.

I'll now describe a basic method of experimental computer science, and sketch how this method was applied to the generation of the VLSI design and implementation methodologies. Later I'll describe the resources required in order to direct this sort of large scale, experimental evolution of engineering knowledge and design practices.

4.1. Experimental Method

There is a basic experimental method that is used in experimental computer science when we are exploring the space of what it is possible to create. The method is especially applicable when creating computer languages, operating systems, and various kinds of computing environments, i.e., applications where we provide primitives that many other people will use to generate larger constructs. Suppose that you've conceived of a new system concept, and want to try it out experimentally. The method is simple: You build a prototype of a system embodying that concept, run the system, and observe it in operation. You might immediately decide, "Hey, this is just not feasible", and scrap the idea right there; or you may think, "Well, maybe we can improve things," or, "Let's try something slightly different," make some revisions, and run the system again. This simple, iterative procedure is sketched in Fig. 6. After the experimentation has generated sufficient knowledge (for example, has demonstrated the feasibility of the concept), you may take a transition into some later phase in the evolution of the concept.

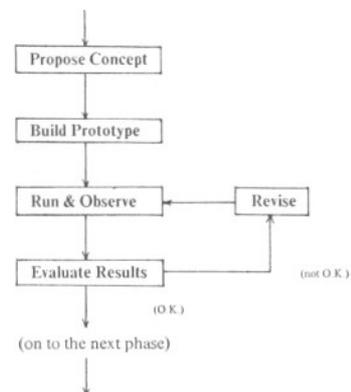


Fig. 6. An Experimental Method.

What might such later phases be? Suppose you've successfully take a new concept through a feasibility test, perhaps experimenting with a quick implementation that you ran yourself. You may think, "Well, let's build an improved prototype, and have some other user run it. I'll watch the user use it, and see what happens." After going around that loop a few times, and making further refinements, you may make the transition to building a prototype to be placed into extensive field trials by many users. Thinking back, you can see how the design course was taken through a succession of such phases, from feasibility to transfer to a few other "users" and on to full scale "field trials". By obtaining feedback from users and observing results at each step, you move on to on the next phase (see Fig. 7) of refinement and integration of that particular system.

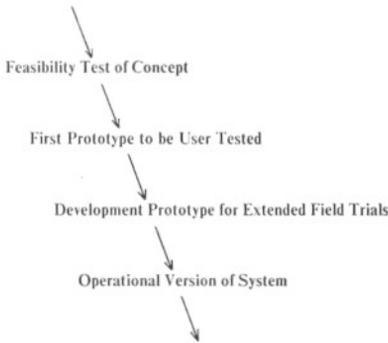


Fig. 7. Some Phases in the Evolution of a System.

If we study the development of the VLSI design methodology, its validation, and its social propagation, you'll notice that the following has happened. The evolution of the methodology involved a multilevel cluster of systems that were being jointly evolved (see Fig. 8). Each system in the cluster runs through the experimental loops, and passes through the various phases of its own evolution. Entries at the higher levels, for example the methodology, or the text, or the documents to support a course, might be more solid and in later phases of their evolution at any given time than, for example, a course in a particular school, or the design environment for that course.

Student design projects play a key role in this

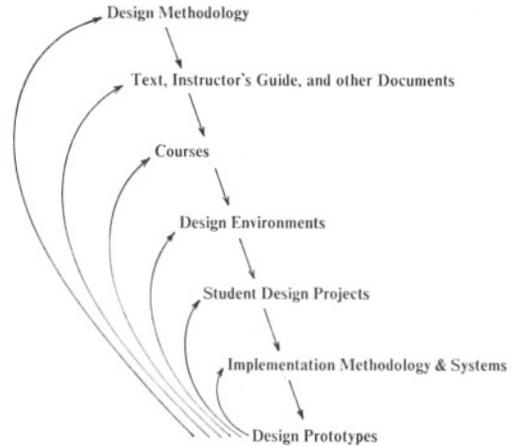


Fig. 8. The Joint Evolution of the Multi-Level Cluster of Systems.

process, supporting new refinements in the higher level systems in the hierarchy every new school semester. Fast turnaround implementation of designs was used to close the experimental loop on all the systems in this hierarchy.

If we think back over the evolution of these systems, we can see how all these things were running in parallel in a rapidly enlarging social enterprise. The early courses run here at Caltech demonstrated that it might be feasible to create a simple design methodology. Following the period of basic design methodology research, the preliminary courses run at Caltech, U.C. Berkeley, and CMU helped debug the emerging text documenting the new design methods. The newly documented methodology was then introduced into the M.I.T. '78 course, which became the prototype for the new type of intensive, project-oriented courses. The results of that course prepared the way for seeding similar courses in many other schools.

The text itself passed through drafts, became a manuscript, went on to become a published text. Design environments evolved from primitive CIF editors and CIF plotting software on to include all sorts of advanced symbolic layout generators and analysis aids. Some new architectural paradigms have begun to similarly evolve. An example is the series of designs produced by the OM project here at Caltech. At MIT there has been the work on

evolving the LISP microprocessors [3, 10]. At Stanford, Jim Clark's prototype geometry engine, done as a project for MPC79, has gone on to become the basis of a very powerful graphics processing system architecture [9], involving a later iteration of this prototype plus new work by Marc Hannah on an image memory processor [20].

While these things were evolving, Dick Lyon undertook the important work of developing, debugging, and evolving a set of basic library cells (see Refs. [2, 5]) that would later be used in all of the courses by all of the students in the MPC adventures. Again, in parallel with that, there was the iterative evolution through a series of experiments, from the early multiproject chip sets to the remote entry multiproject chip done at MIT, to the early implementation systems at PARC, and now on to the automated implementation systems at PARC and USC-ISI.

One thing to remember about this is that such enterprises are organized at the meta-level of research methodology and social organization; they are not planned in fully-instantiated detail using some sort of PERT chart. The evolution of a system of knowledge has a certain dynamics. There is a great deal that happens concurrently. There is the necessity for various activities to reach some minimum sufficient stage of development in order to support activity at some other level. If things are staged right, and people are in close contact with each other and are highly motivated by effective leadership, then a lot of these things can move rapidly forward together. But remember, there is always a strong element of chance when folks go off exploring. The unfolding of the events depends upon what is discovered, and upon how well the opportunities presented by discoveries are seized upon and exploited by the overall community of explorers.

4.2. The Network Community

Some key resources are required in order to organize such an enterprise. Perhaps the most important capital resource that we drew upon the computer-communications network, including the communications facilities made available by the ARPAnet, and the computing facilities connected

to the ARPAnet at PARC and at various universities. Such a computer-communication network is a really key resource for conducting rapid, large scale, interactive experimental studies.

The networks enable rapid diffusion of knowledge through a large community because of their high branching ratios, short time-constants, and flexibility of social structuring; any participant can broadcast a message to a large number of other people very quickly. It isn't like the phone, where the more people you try to contact, the more time-overhead is added so that you start spending all of your time trying to get your messages around instead of going on and doing something new.

The high social branching ratios and short communications time constants of the networks also make possible the interactive modifications of the systems, all of these systems, while they are running under test. If someone running a course, or doing a design, or creating a design environment has a problem, if they find a bug in the next or the design method, they can broadcast a message to the folks who are leading that particular aspect for the adventure and say, "Hey! I've found a problem." The leaders can then go off and think, "Well, my God! How are we going to handle this?" When they've come up with some solution, they can broadcast it through the network to the relevant people. Thus they can modify the operation of a large, experimental, multi-person, social-technical system while it is under test. They don't have to run everything through the completion, and then start all over again, in order to handle contingencies. This is a subtle but tremendously important function performed by the network, and is similar to having an interactive run-time environment when creating and debugging complex software systems.

There is another thing that happens in the network: it's relatively easy to get people to agree to standards of various kinds, if the standards enable access to interesting servers and services. For example, CIF became a *de facto* standard for design layout interchange because we at PARC said "if you send a CIF file to us we will implement your project". Everybody put their designs in CIF!

We answered our own questions: "Is CIF documented well enough to be propagated around?"

Does it really work anyway? Does it have the machine independence we've tried for?" That way we debugged CIF and culturally integrated CIF.

Such networks enable large, geographically dispersed groups of people to function as a tightly-knit research and development community. New forms of competitive-collaborative practices are enabled by the networks. The network provides the opportunity for rapid accumulation of sharable knowledge. Much of what goes on is captured electronically – designs, library cells, records of what has happened in the message traffic, design-aid software and knowledge – all can be captured in machine representable form, and can be easily propagated and shared.

One reason for the rapid design-environment development during '79-'80 was a high degree of collaboration among the schools. Often, as useful new design aids were created, they were quickly shared. Many of the schools had similar computing environments, and the useful knew knowledge diffused rapidly via the ARPAnet.

Another reason for rapid progress was keen competition among the schools and among individual participants. The schools shared a common VLSI design culture; during '79-'80 all used the same implementation system, and batches of projects from the schools were often implemented simultaneously. Therefore, project creation, innovations in system architecture, and innovations in design aids at each of the schools were quite visible to the others. Students and researchers at MIT, Stanford, Caltech, CMU, U.C. Berkeley, etc., could visualize the state of the art of each other's stuff. These factors stimulated competition, which led to many ambitious, innovative projects.

Successful completion of designs, and thus participation in such competition, depended strongly on the quality of the design environment in each school. Therefore, there was strong pressure in each school to have the latest, most complete set of design aids. This pressure tended to counter any "not invented here" opposition to importing new ideas or standards. The forces for collaboration and for competition were thus coupled in a positive way, and there was "gain in the system".

Now, think back to the question, "How do *unsound methods* become *sound methods*?" Re-

member, you need large scale use of methods to validate them, and to produce the paradigm shifts so that the methods will be culturally integrated. In industry, it's very difficult to take some new proposed technique for doing things and put it in use in a large scale in any one place; a manager trying such things would be accused of using unsound methods. However, in the universities, especially in graduate courses in the major research universities, you have a chance to experiment in a way you might not in industry, a way to get a lot of folks to try out your new methods.

A final note about our methods: The major human resources applied in all of these adventures were faculty members, researchers, and students in the universities. The research of the VLSI System Design Area has often involved the experimental introduction and debugging of new technical and procedural techniques by using the networks to interact with these folks in the universities. These resources and methods were applied on a very large scale in the MPC adventures. There are risks associated with presenting undebugged technology and methods to a large group of students. However, we have found the universities eager to run these risks with us. It is exciting, and I believe that it is appropriate for university students to be at the forefront, sharing in the adventure of creating and applying new knowledge. The student designers in the MPC adventures not only had their projects implemented, but also had the satisfaction of being part of a larger experimental effort that would impact industry-wide procedures.

These experiences suggest opportunities and provide a script for university-government-industry collaboration in developing new design methodologies and new supporting infrastructure in many areas of engineering design. The universities can provide the experimental and intellectual arena; government can provide infrastructure and university research funding; industry can provide knowledge about and access to modern, expensive, capital equipment that can implement experimental designs created by university students and researchers. Modern computer-communications networks, properly used, can tie all these activities together. The implementation of designs closes all the experimental loops.

5. Looking Ahead

I wonder where we might apply some of these methods next? Where might people apply methods like these in order to aggressively explore new areas? Well, first of all, there certainly are tremendous opportunities for further discoveries and evolutionary progress in VLSI design and implementation methodology.

We are now seeing the beginnings of new architectural methodologies appropriate for VLSI in a number of specialized areas of application. For example you might study the work that Dick Lyon is doing to create a new architectural set of "VLSI building blocks" for bit-serial digital signal processing [21]. Wouldn't it be interesting if those techniques could now be tried in a few courses? We'd find out if people can really learn about signal processing with VLSI, and then quickly compose working systems, thus providing a reality test of Dick's ideas.

There are many other areas of digital system architecture ripe for the introduction of new architectural methodologies appropriate for VLSI. There are areas like computer graphics for providing high-bandwidth visual displays for interactive personal computing systems, and the generation of computer images for electronic printing and plotting. There's image processing, taking digitized input image data and processing it to recognize and detect things, with applications in OCR systems, visual input systems for controlling robots, smart visual sensors for various defense systems, that sort of thing. There are areas like data encryption and decryption. So there's a whole world of specialized architectural areas that people can now explore, given that they have access to a VLSI design environment and to quick turnaround implementation to try out their ideas. As successes accumulate, the underlying knowledge and the detailed design files can be rapidly propagated around the VLSI network community.

There are many opportunities for evolving new design and analysis aids appropriate for the new design methodology. Progress has been rapid so far [19], but there is plenty more to do. Those interested in creating and testing new design aids might ask yourselves "What can I create and then

introduce over the network that would be valuable to the VLSI community, that might integrate with the overall activity?" That line of thinking, taking into account the current state of the community, and the means of introducing new ideas into the community for testing and validation, may increase your chances of successfully creating something that becomes culturally integrated.

For example, the early circuit extractor work done by Clark Baker [16] at MIT became very widely known because Clark made access to the program available to a number of people in the network community. From Clark's viewpoint, this further tested the program and validated the concepts involved. But Clark's use of the network made many, many people aware of what the concept was about. The extractor proved so useful that knowledge about it propagated very rapidly through the community. (Another factor may have been the clever and often bizarre error-messages that Clark's program generated when it found an error in a user's design!)

Another area of opportunity is in the evolution of standards. For example, we need a standard "process test chip" for the back-end foundry interface, so that designers and foundry operators will have a mechanism for deciding to shake hands and exchange dollars for wafers. Although some strawman versions have been proposed, there is no standard now. Perhaps a standard process test chip could be evolved by inserting strawman versions into wafers that are run for university multiproject chips sets. The community could then gradually converge on a workable standard.

There are opportunities for further evolution of implementation systems. Also, similar design and implementation methods could be mapped into technologies other than nMOS. Design primitives, design rules, and design examples could be created, for example, for CMOS and then run through the same kind of scenario as above to introduce those into a university community.

I myself have become interested in the prospects for bringing about a convergence of the work in VLSI design methodology with work based in knowledge engineering [22, 23]. There is the possibility of creating knowledge-based expert systems to aid VLSI system designers. I can

imagine directing the evolution of such expert systems by using similar methods to those described above: trying out ideas, prototyping them, evaluating them, and bringing them in large-scale use within a computer-communication network community. But an added twist is possible here, that of making knowledge about expert systems accessible to the larger CS community, a community now knowing about VLSI. That way we could help to generate a common literacy about knowledge, a common knowledge representation language, and knowledge about the methods of knowledge engineering.

You'll note that the experimental methods described in this talk aren't limited to application in the exploration of microelectronic system design. I find it fascinating to think about applying these methods to the rapid exploration of other domains of engineering design that may be operating under new constraints, and thus be full of new opportunities.

For example, it is becoming common in some industrial environments for folks to do mechanical system design by using computers to specify the shape and dimensions of parts and to generate the tapes for numerically controlled machine tools that can implement the parts. Consider the opportunity here: What if we documented a simple design method for creating mechanical systems under the assumption that the parts are to be remotely machined and assembled in some sort of "magical automatic factory". Then ask the question, "Well, how would you teach mechanical design under the many new constraints imposed by the remote factory?" If you had access to such a factory, or if you could even emulate it using manual procedures where necessary, you could put in place the same sort of overall experimental environment to develop from very clearly crude principles some sort of new design methodology that would be appropriate for that environment. In that way one could evolve an entire design culture of methods, courses, design examples, design aids, etc. using the methods described above, and that culture could be rapidly spread out through the networks into a large university community.

6. Acknowledgements and Conclusions

I am deeply indebted to many people for their contributions and help in creating the design methods, the textbook, and the implementation methods and system, and also the university VLSI design courses, design environments, and research programs. There are literally hundreds of people who have played important roles in the overall activity. Students, researchers, and faculty members in the universities, and a number of industrial researchers, industrial research managers, and government research program managers have been actively involved in these events. I am at a loss to acknowledge all of the individual participants.

However, I would like to individually acknowledge some of the folks at PARC who've worked on this research since the early days. I am thinking of Doug Fairbairn, who was with us during the key early years; Dick Lyon, who has contributed so much to the effort; Alan Bell and Martin Newell for their innovations and efforts in the creation of VLSI implementation systems that have supported so well the validation and spread of VLSI knowledge. I'd especially like to acknowledge the support and encouragement that all of us at PARC have received over the years from the senior research management of Xerox Corporation, in particular, from Bert Sutherland.

Let's look at the photo of Alan Bell again (Fig. 3), and think back to the MPC79 effort. I'm sure you now sense that MPC79 was not just a technical effort, that there was a tremendous human dimension to the project. So many folks were simultaneously creating and trying out things: students and researchers trying out new designs that were very, very important to them; instructors and project lab coordinators trying out the new courses and project lab facilities; at PARC the new implementation system was coming into existence, under the pressure of trying to provide VLSI implementation service to the many university designers. This built up into a tremendously exciting experience for all participants, a giant network adventure that climaxed as the design-cutoff time approaches, and the final rush of design files flowed through the ARPAnet to PARC.

So when you see someone interacting with a per-

sonal computer connected to a network, rather than jumping to the conclusion that you are observing a reclusive hacker running an obscure program, you might ask yourself "I wonder that adventures this person is involved in?" Remember, you may be observing a creatively behaving individual who is participating in, or perhaps even leading, some great adventure out in the network!

These events are reminiscent of the pervasive effects of the telegraph and the railroads, as they spread out everywhere during the nineteenth century, providing an infrastructure people can use to go on adventures, to go exploring, and to send back news of what they had found. I think of personal computers and the computer-communication networks as a similar sort of infrastructure, but here and now, as we explore the modern frontier – the frontier of what we can create.

The new knowledge and products our VLSI design community is creating will have tremendous social impact, by helping rapidly spread and increasing the power of the new personal computing and computer-communication infrastructure.

References

- [1] C. Mead and L. Conway, *Introduction to VLSI Systems*. Limited printings of prepublication drafts of a text in preparation, Xerox Palo Alto Research Center (PARC), Palo Alto, CA; (a) Chapters 1–3, September 1977; (b) Chapters 1–5, February 1978; (c) Chapters 1–9, July 1978.
- [2] R. Hon and C. Sequin, *A Guide to LSI Implementation*, Limited Printing, Xerox PARC, September 1978.
- [3] G. Steele, Jr. and G. Sussman, *Design of LISP-Based Processors or, Scheme: A Dielectric LISP or, Finite Memories Considered Harmful or, LAMBDA: The Ultimate Opcode*, AI Memo No. 559, Artificial Intelligence Laboratory, M.I.T., March 1979.
- [4] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, 1980.
- [5] R. Hon and C. Sequin, *A Guide to LSI Implementation, 2nd Ed.*, Xerox PARC Technical Report SSL-79-7, January, 1980.
- [6] L. Conway, *The MIT '78 VLSI System Design Course: A Guidebook for the Instructor of VLSI System Design*, Limited Printing, Xerox PARC, Palo Alto, CA, August 1979.
- [7] D. Fairbairn and R. Lyon, "The Xerox '79 VLSI Systems Design Course", *Xerox PARC Videotapes and Lecture Notes*, Xerox PARC, Palo Alto, CA, February, 1979.
- [8] L. Conway, A. Bell, M. Newell, R. Lyon, R. Pasco, *Implementation Documentation for the MPC79 Multi-University Multiproject Chip-Set*, Xerox PARC Tech. Memorandum, 1 January 1980.
- [9] J. Clark, "A VLSI Geometry Processor for Graphics", *Computer*, Vol. 13, No. 7, July, 1980.
- [10] J. Holloway, G. Steele, Jr., G. Sussman, A. Bell, *The Scheme-79 Chip*, AI Memo No. 559, Artificial Intelligence Laboratory, M.I.T., January 1980.
- [11] L. Conway, A. Bell, M. Newell, "MPC79: The Demonstration-Operation of a Prototype Remote-Entry, Fast Turn-around, VLSI Implementation System", *Conference on Advanced Research in Integrated Circuits, M.I.T.*, January 28–30, 1980.
- [12] L. Conway, A. Bell, M. Newell, "MPC79: A Large-Scale Demonstration of a New Way to Create Systems in Silicon", *LAMBDA, the Magazine of VLSI Design*, Second Quarter, 1980.
- [13] T. Strollo, et al., *Documentation for Participants in the MPC580 Multiproject Chip-set*, Xerox PARC Technical Memorandum, 7 July 1980.
- [14] R. Rivest, "A Description of a Single-Chip Implementation of the RSA Cipher", *LAMBDA, the Magazine of VLSI Design*, Fourth Quarter, 1980.
- [15] D. Fairbairn, R. Mathews, J. Newkirk, et al., *Videotape VLSI Design Course based on the Mead-Conway text "Introduction to VLSI Systems"*, VLSI Technology, Inc. (VTI), Los Gatos, CA, 1980.
- [16] C. Baker and C. Terman, "Tools for Verifying Integrated Circuit Designs", *LAMBDA, the Magazine of VLSI Design*, Fourth Quarter, 1980.
- [17] R. Bryant, "An Algorithm for MOS Logic Simulation", *LAMBDA, the Magazine of VLSI Design*, Fourth Quarter, 1980.
- [18] A. Bell, "The Role of VLSI Implementation Systems in Interfacing the Designer and Fabricator of VLSI Circuits", *Proc. of the International Telecommunications Conference*, Los Angeles, Nov. '80.
- [19] L. Conway, "University Scene", *LAMBDA, the Magazine of VLSI Design*, Fourth Qtr., 1980.
- [20] J. Clark and M. Hanna, "Distributed Processing in a High-Performance Smart Memory", *LAMBDA, the Magazine of VLSI Design*, Fourth Quarter, 1980.
- [21] R. Lyon, "Signal Processing with VLSI", *Limited printings of lecture notes for a "constantly evolving talk"*, Xerox PARC, 1980.
- [22] E. Feigenbaum, "The art of artificial intelligence – Themes and case studies of knowledge engineering," *Proc. of the 1978 National Computer Conference*, AFIPS Press, Montvale, N.J., 1978.
- [23] M. Stefik, et al., "The Architecture of Expert Systems: A Guide to the Organization of Problem-Solving Programs," to appear as Chapter 3 in: F. Hayes-Roth, D. Waterman, D. Lenat, (Eds.), *Building Expert Systems*, (a textbook in preparation).

Suggested Reading References

- H. Simon, *The Sciences of the Artificial*, M.I.T. Press, Cambridge, MA, 1969. (2nd Ed. in Press).
- T. Kuhn, *The Structure of Scientific Revolutions*, 2nd Ed., Univ. of Chicago Press, Chicago, 1970.
- L. Fleck, *Genesis and Development of a Scientific Fact*, F. Bradley and T. Trenn, Translators, T. Trenn and R. Merton, Editors, University of Chicago Press, Chicago, 1979. (Originally published as *Entstehung und Entwicklung einer wissenschaftlichen Tatsache: Einführung in die Lehre vom Denkstil und Denkkollektiv*, Benno Schwabe & Co., Basel, 1935.)
- C. Levi-Strauss, *Mythologiques*. Vols. I-IV, Plon, Paris, 1964, '66, '68, '71.
- H. Garfinkel, *Studies in Ethnomethodology*, Prentice-Hall, Englewood Cliffs, N.J., 1969.
- B. Latour and S. Woolgar, *Laboratory Life: The Social Construction of Scientific Facts*, Vol. 80, Sage Library of Social Research, Sage Publications, Beverly Hills, 1979.
- D. Crane, *Invisible Colleges: Diffusion of Knowledge in Scientific Communities*, Univ. of Chicago Press, Chicago, 1972.
- D. Englebart, R. Watson, J. Norton, *Advanced Intellect-Augmentation Techniques*, Stanford Research Institute, Menlo Park, CA, 1972.
- J. Licklider and A. Veza, "Applications of Information Networks", *Proceedings of the IEEE*, Vol. 66, No. 11, November, 1978.
- J. Lederberg, "Digital Communications and the Conduct of Science: The New Literacy", *Proceedings of the IEEE*, Vol. 66, No. 11, November, 1978.

Lynn Conway earned the B.S. ('62) and M.S.E.E. ('63) at the School of Engineering and Applied Science, Columbia University. She then joined IBM Research, where she made major contributions to the architecture and design of ultra-high performance IBM computing systems. In 1969 Lynn joined Memorex Corporation, where she was system architect and leader of the processor design and microprogramming team for a small business computer system.

In 1973, Lynn joined the Xerox Palo Alto Research Center (PARC), initially conducting research in system architecture for image processing. She then founded the LSI Systems Area, forerunner to the present VLSI System Design Area at PARC, a department responsible for Xerox's Corporation's research program in VLSI system architecture and design methodology. During 1978-'79, Lynn also served as Visiting Associate Professor of EE/CS at M.I.T., organizing and teaching the first VLSI design course there.

Lynn Conway is one of the key innovators of new methods that greatly simplify the design and implementation of micro-electronic systems. These methods, and the infrastructure-building work of Lynn's group at PARC, have played a central role in the rapid establishment of a VLSI systems research community in leading U.S. universities and research organizations, and courses in VLSI design at major universities throughout the world.

Lynn is co-author of the textbook *Introduction of VLSI Systems*. She was appointed a Xerox Research Fellow in 1980, in recognition of her contributions to Computer Science. In 1981, she received the annual *Electronics* Award for Achievement, for her work on VLSI design methods. Her primary research interest at present is knowledge engineering, applied in particular to the deliberate engineering of methodologies for digital system architecture and design.