Proposal for a Joint Effort in Personal Scientific Computing

Carnegie-Mellon University

Department of Computer Science

August 23, 1979

This proposal is being sent to a number of computer manufacturers to solicit their interest in joining with the Carnegie-Mellon University Computer Science Department in building SPICE, a personal scientific computing environment, using hardware that will be commercially available on a large scale.

Circulate + Return to me

1 HF

~~2 Bell D.~~

~~3 Dick~~ got burn 1/14/80

4 ~~BJ~~ ~~FODICK~~ Snyder

5 ~~Dick Snyder~~

6 Wayne Rosing

7 Sproull

file

OCT 9 1979

10-11

JAN 28 1980

Proposal for a Joint Effort in Personal Scientific Computing

Carnegie-Mellon University

Department of Computer Science

Dick: Yep the target! might not believe the cost & date by a few years but the target is right & we must keep VAX pressing for the target. also for the next 10 years we got to make money on stupid relatively fixed applications on low cost smart terminals with moderate comm speeds.

wf: Yes; IIS no longer research - this is the time to make money on the $50M investment we already made!

Snyder: I agree. Today we have TIS providing personal computers (our terminals) connected via memory. Next step is better terminals (SPICE) connected with local area networks

ROSING: THEY FORGOT ABOUT CENTRAL COMPUTER RESOURCES - THIS SYSTEM WILL REQUIRE MORE CENTRAL MIPS/USER. IF THEY PAY 50% OR SO, IT'S A POSSIBLY REASONABLE PROJECT, BUT I DON'T SEE US GETTING A LOT IN RETURN EXCEPT GOOD PRESS + A SHOWCASE. AFTER A FEW YEARS, HOWEVER, NEW APPLICATIONS SHOULD SURFACE - A PRODUCT OF THE NEW "ENVIRONMENT". SIGNIFICANT THOUGH, IS THEIR UNDERSTANDING OF THE ~~USE~~ INTEGRATION of VOICE INTO THE NETWORK.

Table of Contents

Introduction

The era of time-sharing is ending. Time-sharing evolved as a way to provide users with the power of a large interactive computer system at a time when such systems were much too expensive to dedicate to a single individual. The size of these systems made a *qualitative* difference in the kinds of service they provided, and the fact that many users ran on the same machine made possible a level of user cooperation and communication that could not have been achieved on small individual machines.

Recent advances in hardware technology are opening up new possibilities. The level of capital investment which today provides each user with a small slice of a time-shared machine and a crude CRT terminal will, by the mid-1980's, provide that same user with his own powerful machine, far more powerful than today's microprocessors and equipped with such features as high-resolution color graphics and audio I/O devices. This development will enable the user to avoid many of the compromises and limitations inherent in time-sharing. New high-speed network technologies make it possible to move to this personal computing environment without foregoing the attractive features of time-sharing: shared information, good inter-user communication, and the sharing of expensive peripherals. We are entering a new world of versatile personal computers, as different from the world of time-sharing as that world was from batch-processing. It will take years of work at many different installations to learn how best to exploit these new opportunities.

We in the Computer Science Department of Carnegie-Mellon University (CMU-CSD) intend to participate in creating and exploring this new world. A start in the exploration has already been made at Xerox PARC with the ALTO environment and its progeny and at MIT with the Lisp Machine project. Both of these projects are based on specially-designed prototype hardware that will not be available commercially. In creating our own personal computing environment, CMU-CSD could follow a similar route, or we could wait for some established manufacturer to market the appropriate hardware. But we would prefer to follow a third route: we would like to work closely with some computer manufacturer to develop a personal computing environment based on a machine that will be commercially available on a large scale. We are prepared to make a substantial long-term commitment of our time, expertise, and imagination to this partnership; we are looking for a comparable commitment of resources by our partner. We propose to call the resulting environment SPICE, for Scientific Personal Integrated Computing Environment.

A cooperative arrangement has three main advantages for us. First, we avoid the logistical problems associated with designing and building our own hardware. Second, under the proposed partnership we could obtain the hardware for a full-scale SPICE environment sooner than we could otherwise afford. Third, the results of our work are immediately portable to other sites that have or could buy the same equipment. This would allow us to share more effectively programs and techniques developed within this larger community. We believe that the impact of this kind of personal computing can be revolutionary, but the revolution cannot

occur until the hardware becomes generally available through commercial channels.

We believe that our partner in this enterprise will derive great benefits as well. He will be able to draw on the proven experience of CMU-CSD to provide constructive technical advice in the development of a new product which will define the state of the art. In addition, since he will be among the first to develop a personal computing system for the commercial market, he will be uniquely placed to become a leading supplier. His product will create the market and also provide a standard against which potential competitors will be compared.

In the remainder of this paper we will describe in more detail our view of what the personal computing environment could be like, the technical requirements of the hardware and software, and the kind of partnership that we would like to form .

The Vision

Overview

We expect that CMU-CSD in the mid-1980's will be a community of about 200 researchers engaged in research and development activities in all parts of computer science: programming systems, computer architectures, VLSI design, data bases, computer networks, human-computer interaction, artificial intelligence, theory of computation, etc. This diverse community will be served by a common computing environment which will provide an ever-growing collection of software tools and capabilities. In addition to providing facilities for computer science research, the environment must also provide a number of office-automation services: report preparation, personal communication, management of meeting schedules, maintenance of departmental records (some of which are confidential), etc.

The environment we envision has the following components. (More technical details and some rationale for these requirements will be provided in a later section.)

1. **A set of personal computers**, roughly one per researcher, each with something like the power of a medium-sized time-shared computer of today. We see a need for the following features in these machines:

Speed:	10^6 macro-instructions/second.
Control store:	Room for at least 16K micro-instructions, reasonably structured and user-writable.
Virtual address:	Provision for 2^{30} to 2^{32} bytes of address space, smoothly addressable.
Primary memory:	1 Mbyte, minimum.
Secondary storage:	100 Mbytes, suitable for paging.
Display:	Color, bit-mapped, high resolution (1K x 1K pixels); with a good keyboard and a "mouse" or other pointing device.
Other:	Transducers and digital converters for audio input and output. Provision for attaching a TV camera or other device for input of static images.

   This describes the *typical* configuration; it must be possible to add additional main memory, control memory, secondary storage, special processing boxes, and peripherals as needed. The architecture must be sufficiently powerful and general to assure a long, stable lifetime, since our investment in the software will be considerable. We would expect that by the mid-1980's such systems could be priced around $10,000, though they might cost many times that figure today.

2. **A high-bandwidth network** connecting all of the processors in the community, including some experimental processors and special-purpose devices.

   In an environment such as ours, it is very important that the network be able to accommodate alien, highly specialized devices and interfaces to other networks in addition to providing efficient service to the standard SPICE machines. We expect to develop modes of operation in which larger tasks can be divided into parts and passed out to idle processors on the network. This local network will have reasonably transparent connections to comparable networks at other installations around the country, even if these sites use hardware that is different from ours.

3. **A shared file system** to store information for all users on the network.

   This file system may run on one central file computer or it may be distributed around the network, but it will appear to users as a single unified facility. This repository will promote

sharing of information within the community, and will make it possible to keep all files synchronized and up to date. Backup copies will be made automatically, and great care will be taken to ensure that this system is never unavailable for long. A large library of reference material will be available on-line using video disk or other comparable technology.

4. **Shared devices**, such as hard-copy printers, on the network.

   We would expect to have at least one high volume, high resolution xerographic printer for most output and a slower phototypesetting device for book-quality output. Other shared facilities on the network might include computer controlled wire-wrap and some large-screen displays for classroom use.

5. **The SPICE software environment**, which will be as important as the hardware.

   Among the facilities that must be provided are a comprehensive editing and text-processing system; mail, message-handling, and other forms of inter-user communication; a system for sending reminders and scheduling the user's appointments; extensive program development facilities in several high level languages (creation, debugging, analysis, management of programs, and library maintenance); an integrated graphics-oriented hardware design facility; and general-purpose tools such as calculators, symbolic mathematics packages, personal data-base systems, and so on. All of these systems should be easy to use, should have online documentation and tutorials, and should share command conventions wherever this is reasonable.

Note that the integrated, communication-rich environment described here differs in significant ways from the vision usually associated with the term "personal computing." The users of SPICE will form a community that shares programs and data structures, builds facilities for the common good, works on coordinated projects from many machines, and sometimes distributes a single computation among many machines. The substantial power of each machine assures that most of a researcher's work can be done within the responsive SPICE environment. The users will live on the SPICE machines, both for their technical needs and for the mechanics of everyday life: arranging meetings, communicating with other users, preparing documents, etc. None of this is characteristic of today's low-power microprocessor systems.

Advantages of a Personal Machine

Computer hardware becomes cheaper every year, while skilled manpower becomes ever more expensive and scarce. It is sound economics to provide computer users with personal computing facilities to increase their efficiency and productivity. We must stop worrying about idle machine cycles and start learning to use those cycles to benefit the user. Personal computing offers a number of advantages over a time-sharing system:

1. **Large, cycle-intensive programs** can be run efficiently.

   The scheduling and storage maintenance strategies on each machine can be tailored to meet the current needs of that machine's single user. This means that a large virtual memory system can be implemented with minimal contention and overhead, using the paging storage local to each machine. Any task can be given the machine's full resources, without regard for the needs of other users. There will be no more painful efforts to get around memory limitations and no more waiting for a large job to be swapped in.

2. **Accessible microstore** which allows the user to tailor the virtual machine to his particular task.

   In the Lisp Machine, for example, most of the Lisp run-time system is written in microcode, with room left over for the user's critical inner loops. By coding a few critical routines in microcode

the user can often obtain a substantial increase in the overall speed of a program. This may be worthwhile only for programs that will be used heavily, but in these cases it can be extremely important.

3. **High-resolution graphics** of the kind we envision will provide the user with an environment that is qualitatively different from that provided by today's standard CRT terminals.

   The added resolution makes it reasonable to divide the screen into distinct areas, each monitoring a different task. Text to be output can be displayed in exactly the form it will take on the printed page. Graphical and half-tone material can be intermixed with text, and changes in the display can be made rapidly. Color is essential for multi-layer VLSI design and should prove to be quite useful in other contexts as well. Although the CPU is not involved in maintaining static images on the screen, it must be possible to obtain a substantial burst of computation instantly, whenever the display is to be changed. Instant attention is also required if the system is to track a mouse or other pointing device properly. A time-sharing system cannot provide this level of service to all of its users.

4. **Natural communication** will be possible with the audio I/O facilities.

   A microphone, A/D and D/A converters, and a speaker will make it possible to send voice messages, to store these messages in digitized form, to annotate documents with voice, and to enter into voice dialogs with colleagues elsewhere on the network. In addition, the experience gained in the successful speech-understanding project at CMU leads us to believe that it will soon be feasible to use simple voice commands as a flexible and natural way to control programs. The necessary audio devices are now inexpensive and having them available on all machines will help us to evaluate their true potential. As with the graphics displays, these devices require occasional real-time bursts of intensive computing which would be hard to provide on a shared machine.

5. **Full availability** of all the machine's resources, independent of what other users are doing.

   No longer are big jobs forced to run at 4 a.m. No longer do simple edits become intolerably slow in the afternoon. Each user will get the same excellent level of service. Only the shared facilities and perhaps the pool of supplemental computing resources are affected by the overall system load.

6. **Reliability** is an inherent part of the environment.

   A failure in any one machine cannot bring down the others. The critical demonstration or production run can simply be moved to a working machine. The only critical resources are the central file system and perhaps the printer. These must be designed for extreme reliability or backups must be available.

7. **Extending the environment** can be done in smaller increments than in a comparable time-shared system.

   If a few new users join an installation, a few new SPICE machines can be bought for them. There will be no more agonizing decisions between running with one overloaded machine or two underutilized ones. The environment can be closely tailored to the community's needs. An important corollary of this is that an installation with only a few users can obtain essentially the same computing environment for these people that the larger institutions provide for their hundreds of users.

Some Scenarios

It is hard to describe in abstract terms how one computing environment differs from another. Many subtle differences can only be appreciated when they have been experienced first-hand. Since we are unable to convey such experience on paper, we will have to describe the feel of the SPICE world by way of some imaginary scenarios.

Scenario 1: Document Production

A user is writing a technical report at his machine. With a truly responsive text editor, it is easier for most users to compose their text on-line than to use more conventional means. The resulting prose tends to be of higher quality, since it takes much less effort to rearrange and alter text on the screen than to erase or cut and paste. As the user works, the machine can call up dictionary and other reference information and show him as he types how the finished text will look. The computer can check spelling and certain simple grammatical rules and alert the writer to problems. Illustrations and graphs can be created along the way or can be called up from the archives. To add an illustration to his document, the user may scan in an existing illustration and "paste" it into the text. When the text is completed it can be sent at once to the central repository, from which anyone on the net (or other connected nets) can access it. It can also go to the photo-typesetter for creation of hard copy.

Scenario 2: Program Development

A programmer is creating his program online. The editing program knows a good deal about the syntax of the language at hand and prevents non-syntactic constructions (unbalanced parentheses, for example) from being entered or indicates such dubious constructs with color-coding. As the program is typed in, it is automatically pretty-printed so that it always appears in the form most understandable to the programmer. Manuals, project specifications, instructions for using the editor, and an extensive library of algorithms, programs, and subroutines are all just a few keystrokes away. As a module is completed, it is passed to a project management system which coordinates the versions and specifications for each module and which remembers what tasks have yet to be done.

Scenario 3: Program Debugging

Now the program is being debugged. As the user single-steps the program, the corresponding source code is visible in one portion of his screen, with the statement being executed highlighted in color. The current stack of procedure calls appears in a second window (in a format corresponding to the source code) and the values of a dozen or so variables are being continuously displayed in a third window. As the source of a problem is discovered, it can be corrected in the source file, and the fix is reflected at once in the running image. If the external behavior of a module must be altered, an indexing system can tell the user what other modules are made inconsistent by the change. At any time, a user can get a list of inconsistencies yet to be resolved. Environments close to this have been built around the LISP language; we hope to develop comparable environments for algebraic languages such as Ada and for systems whose modules are written in several languages.

Scenario 4: VLSI Design

A logic designer is working on a new VLSI chip. As he draws the layout on his color screen, using a mouse and menu system, a constraint checking program is monitoring his work, predicting performance, adjusting the size of traces, and so on. This computation may be running entirely on the local machine or parts of it may be sent off to run on unused processors or other computing resources in the network. As he works, the designer is able to call up cookbook design elements from the on-line reference library and, if these meet his needs, plug them into the current design at desired locations. As this occurs, the constraint checker adjusts the parameters of the new element to fit the given context. When the designer is satisfied, he runs a final simulation, then transmits the design to the automated mask-making and chip-fabrication facility. A very similar scenario could be written for designing bridges, buildings, aircraft, or whatever.

Scenario 5: Multi-media Communication

Several researchers are cooperating on a large project. Each researcher is working in a different building on his own part of the project and is using his own machine. From time to time the researchers need to confer, in pairs or all together. Any user can initiate a linkup, and the others can accept or reject it. Linked users can talk to one another using their terminals' acoustic I/O and compressed digital transmission. Alternatively, they can type or sketch on their linked-together screens. If something important is said or sketched, a user can save it by sending it to his personal file of memos. When the link is terminated, the users can restore their screens and proceed with what they were doing; there is no loss of context. The same mixture of speech, text, and sketches can be sent to another user (or to everyone on a mailing list) as mail. They will see it the next time they log in or check their mailbox files.

Scenario 6: Non-Intrusive Communication

A faculty member is working on a paper, rushing to meet a deadline. He does not want to be interrupted by any but the most critical messages. Some messages must be let through, however: an expected call from a major funding agency, queries from a student preparing a companion paper, calls from the professor's spouse. The list of people and topics to be admitted is stored on the machine and compared against each incoming message. Qualifying messages are passed through at once; others are stored as mail, with the senders being notified of the situation. Optionally, some senders and topics can be ignored altogether. The topics are determined by a hierarchical scheme based on categories of keywords -- an exact match is not required. When the user finally finishes his paper, he can respond to the queued requests by mail, by linking terminals if the sender is logged in anywhere in the country, or by phone. The computer does the searching, linking, and phone dialing. All of this could, of course, be done by a competent secretary, but good secretaries are expensive, scarce, and usually work only during normal office hours.

Scenario 7: A Telephone Answering System

A researcher is working after hours at a machine far from his office. An important phone call arrives. After three rings, the phone is answered by one of the SPICE machines which is acting as telephone operator. A recorded message asks the caller to state the name of the person being sought. A speech recognition program analyzes the spoken name and locates it on a list of the system's users. The recipient is located, and the call is patched through to his terminal. If the user is not available or declines to take the call, the operator machine allows the caller to dictate a message that will be saved in audio form (compressed digital encoding). This will be sent to the intended recipient as mail. The kind of speech recognition required for identifying names from a list of a few hundred is within our current capabilities, though it is computationally expensive on conventional machines.

Scenario 8: Information Distribution

A new member has joined the department. When he first logs in to the system, the newcomer is asked for a variety of biographical information. He also has his picture taken by a television camera tied to one of the machines. The next day a message introducing the new person and displaying his picture appears among the system mail of all users. This information also goes to the file system. Whenever anyone wonders who John Smith is, the picture and biographical information can be instantly displayed on the screen of any SPICE machine.

The Design

The following is an outline of the design, which will become more specific after consultation with the partner.

A great deal of the background for the design comes from the example set by the Xerox Alto and its related software. CMU-CSD is extremely fortunate to have been given, in a generous grant from Xerox, a number of these prototype systems: several Altos, a central file system, and a Dover raster-scanned printer. Our design for the SPICE hardware and software borrows heavily from the experience of the Xerox experiment.

Almost all of the components of SPICE have been previously tried in some form; we are proposing a unique combination, but we are not venturing into completely unknown areas.

Hardware

We know of no computer presently available commercially that exactly meets the needs of SPICE, though some predecessors exist such as the MIT Lisp Machine and the Xerox Alto. Moreover, at least one new design has many of the characteristics we seek: the PERQ system designed by the Three Rivers Computer Corporation.

To reiterate the requirements listed earlier, a SPICE machine available in 1985 should have roughly the following characteristics and features:

1 MIPS execution rate
User-writable microstore
Large virtual address space
1 Mbyte primary memory
100 Mbytes secondary memory

High quality color raster display
Keyboard and pointing device

Audio input-output
Provision for connecting image input
High bandwidth network connection
External bus(es) for auxiliary devices

Packaged for office or laboratory use
$10,000 price (mid '80's)

These numbers are intended to be indicative of scale and power, not of precise features. Each of these objectives receives further comment in the paragraphs that follow.

Instruction Execution and Microcode

One of the main objectives of the basic SPICE processor must be to offer long life to SPICE software, since the investment in this software will be large. In considering this problem, we may assume that SPICE will be written almost exclusively in a high-level language.

The traditional approach to this problem is to use a standard, fixed instruction-set architecture that can be sensibly implemented with a range of performances over a number of years. For example, the IBM 360/370 architecture has this property. At the other extreme, the 8080 design was technology-induced, and is already obsolete. Because the SPICE computer will be programmed using one or more high level languages, minute details of the machine's instruction set are not terribly critical. Generally, it should allow straightforward compilation of moderately efficient code for languages such as PASCAL or Ada.

We prefer a design that harnesses user-writable microcode for the purpose of increasing software life. Several benefits accrue from the ability to write new microcode and thus modify the virtual machine seen by the programmer:

1. A powerful macro-instruction set can be implemented cheaply and in such a way that changes to the underlying technology can be made invisible to the software. This insulates the instruction set from pressures due to short-term technological constraints of the sort that have led to the grotesque instruction sets of many current microprocessors.
2. Minor extensions and modifications to the macro-instruction set can be permitted as new needs are perceived. This adds to the useful lifetime of the instruction set.
3. Customized virtual machines can be created which are specialized for different languages or tasks (e.g. to create a virtual LISP or PASCAL machine). This approach increases the efficiency of interpreted languages (relative to interpreters written in macro-code), and relieves a compiler from generating code for a machine ill-suited for the language.
4. Small "kernel" or inner-loop programs can be implemented in a very efficient way. Placing just a few critical routines in the microcode can make a dramatic difference in the execution time of some programs. For example, the operations to manipulate data in the frame-buffer display are obvious candidates for this treatment.

We feel that benefits (2)-(4) are sufficiently important that the presence of a sizable user-writable control store is essential to the success of SPICE. All uses of the microcode require that the micromachine be designed with programming in mind, and that it have a reasonably clean and complete structure. Using microcode for "kernels" requires convenient ways to invoke the kernels from macro-instructions (e.g., using unimplemented operations or traps). Using the microcode to implement language-dependent virtual machines requires several aids in the micromachine (e.g., instruction byte fetching for a byte-coded instruction set such as PASCAL PCODE). Note that the use of microcode to ease migration to new hardware, to build kernel loops, and to build specialized virtual machines is rather different from the use envisioned by most computer manufacturers, to reduce the implementation cost of a particular (fixed) instruction set.

Performance

The performance objective of 1 MIPS is, of course, a vague one. We have in mind a "typical" PASCAL statement such as **c := a+b** compiling into two instructions, each taking 1 microsecond or less.

It would be a mistake to concentrate entirely on the user interface made possible by SPICE; at least as important to us, and to the future of personal computing, is the potential of personal machines to increase the effective computing power available to each user. SPICE must be an effective computing environment for

advanced computer research and computer-aided design, not merely a collection of smart terminals.

Virtual address and memory size

Our view of personal scientific computing is not limited to small, well-bounded applications. The size of programs written by designers and researchers is large and growing. The size of a programming environment that comfortably supports a researcher is large; in addition, the SPICE vision anticipates a qualitative and quantitative increase in the amount of information summoned to aid the researcher: manuals, electronic mail, working documents, etc.

The virtual address space must be large. CMU-CSD has long ago outgrown 16-bit address spaces (PDP-11's) and 18-bit address spaces (PDP-10's). Our struggles with small addresses have slowed work in multiprocessors, operating systems, image understanding, symbolic computing, and artificial intelligence. We believe a move to a 24-bit address space is merely a stopgap (especially if the unit addressed is a byte rather than a word), but that 30 or 32 bits should be sufficient for a long time. A recent ARPA-sponsored workshop on facilities for symbolic computing seemed to achieve consensus on this point.

Sizes of primary and secondary memory must be reasonably large to match a large virtual memory:

virtual address (2^{30} bytes)	10^9 bytes
secondary memory	10^8 bytes
primary memory	10^6 bytes

Once again, these numbers are for the typical SPICE machine. The hardware must allow for some machines to have substantially more primary and secondary memory than these minimum figures suggest. We believe that addressing down to the byte level is desirable. Greater precision in the numbers in the table is pointless in the absence of a specific processor design.

Display

The raster display is a mandatory part of the SPICE computer; display output is used extensively by all SPICE software, not just by one or two "graphics programs." We believe the display's essential features are:

1. **High resolution.** To approximate a display as complex as the information on a typical 8 1/2 x 11 inch page, something like 600 x 800 bi-level (black or white) pixels are needed; more are desirable. The CPT 60 Hz monitor is adequate in terms of resolution. (To some extent, spatial resolution can be exchanged for gray-scale resolution.)
2. **Frame buffer (bitmap) representation.** The frame buffer is the only image representation technique that is completely insensitive to image complexity. Arbitrary images can be displayed by setting the intensity of each pixel in the frame buffer.
3. **Cursor.** Some provision needs to be made for a cursor. A good solution is a small bitmap buffer that can be read and written by a program just like the main frame buffer. In addition, the (x,y) position of the image of the small bitmap can be changed easily.
4. **Image-generation aids.** Some operations on the frame buffer are sufficiently tedious and simple that they deserve hardware and/or microcode assist. Examples are vector generation, character generation, and RasterOp (See Newman and Sproull, *Principles of Interactive Computer Graphics,*

2nd Edition, p. 264). We believe that a fast RasterOp implementation is especially important.

5. **Access to the frame buffer.** It is essential that an application program have direct access to the frame buffer, i.e., that pixel values can be read and written by ordinary memory references. This facility is needed because hardware image-generation aids cannot anticipate *all* uses of the frame buffer. With direct memory access, an application program can achieve any effect desired, although perhaps slowly. A corollary of this is that RasterOp must have access to the application program's memory to copy bitmaps in and out of the frame buffer and to find font representations. Generally, this requirement means that the underlying SPICE computer must have substantial memory bandwidths, so that display refreshing and instruction execution can both proceed at high speed.

6. **Color.** The SPICE computer must support color displays, perhaps as an option. A modest approach is to describe each pixel with 4 bits and to use a map to convert these values into 16 arbitrary colors for output. More elaborate color displays (up to 30 bits/pixel) will have significant application in imaging work, and might also be considered as an option.

7. **Provision for video input.** Some applications, such as interactive conferencing and document processing, can make very effective use of television or facsimile input. This facility is most easily combined with the frame-buffer display: video for a single frame is sampled and stored in the frame buffer, which can then be read by a program at modest rates. This technique will not permit the recording of real-time moving full-frame images because there is not likely to be sufficient processor or disk bandwidth to save the contents of the frame buffer before another frame must be stored.

8. **Expansion.** It is desirable to have the optional features easily added by plugging modules into a standard cage. Thus, the video input and color options will not require wholly different machines, but will be additions to the standard SPICE machine. Similarly, it would be desirable if a single SPICE computer could support additional displays, though perhaps with lower performance on the additional screens. The key point here is that the scheme for interfacing the processor to the display must provide for such extensions in a smooth and coherent manner.

Pointing Device

The requirements for a pointing device are difficult to state. It should be comfortable and easy to use. It is entirely independent of the display; it merely reports a two-dimensional position or incremental motion to the processor. Its chief uses will be for pointing to menu items shown on the screen, and for selecting text characters or graphical objects already displayed. For these purposes, it should have some buttons (on the device itself, not on the console) so that a user can indicate "this is the point I mean." Precision and repeatability are not particularly important, but reliability is essential. Something like the SRI mouse is reasonable.

Keyboard

The primary requirements for the keyboard are that it feel good and that it *be reliable*. We find the cheap keyboards on most current character terminals not only unpleasant, but destructive: they unreliably transduce the user's finger strokes. It should be possible for the processor to sense, at any instant, exactly which keys are depressed on the keyboard. The keyboard might report to the computer a bit-vector that marks depressed keys as 1 and idle keys as 0. Most keyboards do not have this property; instead, they "encode" the downward keystrokes into ASCII, modifying the encoding based on a few shift keys. Such keyboards are too

inflexible for an environment in which the nature of the user interface is a subject of active research and evolution. To eliminate polling, the keyboard interface hardware should be able to interrupt the processor whenever the keyboard state changes.

Audio input-output

Standard equipment on the SPICE processor should support audio input-output. A simple solution is digital-to-analog and analog-to-digital converters supporting, say, 8-bit samples at rates from 8000 to 20000 samples/sec. Modest hardware or microcode facilities to permit silence detection, simple digital mixing, and synthesis are desirable. Although we do not envision complex vocoding (e.g., LPC) being necessary, we would like to be able to mix several digital audio streams for output, and to synthesize various tones and noises for interactive dialogues.

Some sort of headset with a mounted microphone should be part of the SPICE computer package.

High bandwidth network connection

The personal SPICE computers must be linked together with high-performance communication. The physical transmission medium should be easy to install and attach to, as it must reach into many offices and laboratories. The major communications are likely to be:

1. Personal communication among researchers
   - Electronic multi-media mail
   - Real-time digitized speech
   - Conferencing

2. Access to services that are centralized for economy
   - File storage
   - Printing
   - Magnetic tape
   - Special I/O (e.g., phototypesetter)

3. Sharing data
   - File storage

4. Interprocess communication, when several processes running on separate computers are organized to address one computational problem.

There are several reasons for supporting these communication needs with a high-bandwidth design. First, a communication system must be shared among many computers; high total capacity is required so that the share available to each pair of communicating processes is adequate. Second, our desire for transparent access to network services requires high speeds; if a file is to be accessed on a remote file system as quickly as on local storage, the network transmission bandwidth must be comparable with disk transfer bandwidths. Third, the media-intensive design of SPICE (voice, video) requires larger communication capacity than for most computer communication networks, which carry little or no high-bandwidth media traffic.

These considerations, coupled with some knowledge of what is currently feasible, lead us to choose a bandwidth of about 10 Mbits/sec. Proper design of the network software will allow SPICE to accommodate higher performance transmission systems later on. Although the software network protocols must be standardized throughout the life of SPICE (else face massive compatibility or re-programming problems), the transmission system can evolve steadily, using "gateways" to previous, slower, transmission systems.

Because the network provides access to essential services, its availability must be extremely high. A low transmission error rate is not essential (though desirable, of course) because we intend to use packet protocols to detect errors and to retransmit packets found to be in error.

External connections

Researchers often want to augment their personal machine in some way to address specific problems. In addition, special I/O gear may be added to a SPICE machine to offer a service accessible to the community via the network (e.g., printers, raster input scanners, film recorders, archival memories). We envision three kinds of external connections:

1. Simple, low bandwidth (e.g., RS 232, IEEE 488, and/or parallel "switch and relay" register), used for telephone-line connections, for small devices such as plotters and impact printers, and for connections to experimental hardware (the parallel interface is especially useful for this last).
2. Addressable, high bandwidth. This bus offers the same addressability to I/O devices that is available to user programs. This allows I/O devices to use DMA for simple purposes (like a channel) or to interpret rather complex data and control structures built by the user program.
3. Processor extensions (optional). Occasionally, an experiment may require specialized computation in sufficient quantity that it is best provided by a minor processor extension. An example is an image compressor/decompressor that transforms one description of an image stored in memory to another one, also stored in memory. Another example might be special hardware added to increase the speed of floating-point operations on those machines that use floating-point a great deal.

Package and Price

Ultimately, the SPICE computer must be packaged for convenient placement in an office or laboratory, with as much mobility as possible. We hope for a mid-80's price of $10,000.

It is likely that neither objective can be met immediately. Interim packaging solutions are necessary, although the user I/O gear (display, keyboard, pointing device, audio) should be packaged for convenient desktop use from the start.

Evolution

If SPICE is to have a long life, we must make some plan for natural evolution of the hardware as technology changes.

We believe the most important consideration is that SPICE hardware must support, from the start, *all those*

functions required in the final version of the system: audio, raster graphics, one machine per person, and substantial computing power. An interim machine that omits, for example, audio facilities, will not induce software designs that exploit the voice medium, no matter how hard we try to plan for its eventual inclusion. if the early hardware has audio input/output features, then all facets of the software environment, from message systems to programming environments, will be designed to include voice in natural ways. By similar argument, an interim machine that does not offer substantial computing power will not become the basis for significant research computing, and a SPICE scientific computing environment will simply not grow in the barren machine.

The size of the address space is a surprisingly fixed part of the SPICE hardware definition. Although it is possible in some machines to extend the address space without incurring major costs, most previous efforts to increase the addressing capabilities of a computer have been accompanied by significant re-programming efforts. A splendid example is the PDP-10, which started with an 18-bit address, and has been extended to 23 bits. The programming environments that make the PDP-10 so valuable (e.g., Interlisp) must be almost entirely rewritten to take advantage of the larger addresses. Since it is easier to build the computer with extended addressing than to adapt the software, it is, therefore, desirable to build the computer with extended addressing from the start.

The choice of an address space of about 30 bits is a compromise between feasibility and caution. Caution suggests that address space requirements grow by one bit every two years, with the current value at about 24 bits. SPICE's value in the research world will thus be limited to about 12 years. Choosing larger addresses, a more conservative approach, tends to require large word sizes as well, leading to performance and cost problems. Clearly, any proposal for SPICE hardware that accommodates virtual addresses larger than 30 bits will be well received.

It is possible to begin SPICE development with a smaller virtual address space than will be needed ultimately, but only if the transition to the ultimate large size has been very carefully planned at the outset and all software is written so that the transition is invisible. We will need something like 100 Mbytes of virtual memory even for the first machines.

The remaining facilities of the machine can be easily increased as technology improves or costs come down. More primary memory, more local storage, more control memory, and faster instruction execution are all desirable. However, any additional facilities must not introduce the need for reprogramming.

To prevent reprogramming costs, some aspects of the hardware must be fixed, even though technological advances will surely allow improvements. The fixed elements are usually termed *architectural* features, and must be supported throughout the life of a family of machines. The instruction set is probably the least critical part of the architecture, because the exclusive use of a high-level language in writing SPICE will allow conversion to a new instruction set simply (!) by modifying the compiler. The architecture of input/output devices is more critical: program structures will depend in detail on how the display is controlled, for example. Surprisingly, even the gray-scale resolution of the display may have profound impact on the software: techniques that suffice for 1-bit-per-point displays are wholly inappropriate for displays that have several gray levels. (Of course, one may achieve "backward compatibility" by operating the multilevel display as if it used only 1 bit per point, but this is presumably undesirable.)

Amendments

The hardware image we have presented is by no means inviolate. We anticipate that our partner in the SPICE venture may wish to shape the design to accommodate other product needs. For example, it is possible that some people may view a SPICE computer as a very capable graphics terminal rather than as a personal computer. The design might make certain features optional (e.g., audio) even though all machines supporting SPICE in our environment would require the feature. We hope that suggestions from the partner will help improve the SPICE hardware capability.

Software

The basic facilities which will be provided by the software are as follows:

1. Scientific (research) programming
2. Document production
3. Communication

Each of these areas covers a wide range of facilities. Programming facilities include languages, runtime systems, debugging facilities, program-construction aids, etc. Document production facilities provide for every type of text, from an informal mail message, through program texts, technical reports, and letters, to manuscripts and books. Facilities will also be included for putting graphical illustrations and voice annotations into documents. Communication is perhaps the broadest category: exchanging mail with colleagues, maintaining calendars, using voice and graphics in messages.

The SPICE system will be driven by a set of objectives that sets it apart from other personal computing endeavors (notably that of Xerox PARC). Some of these objectives are set forth below:

1. SPICE is, first and foremost, a *computing environment.* The SPICE hardware architecture, with its large virtual address space, makes the single biggest contribution to this objective. SPICE will build on this to provide interactive access to programming tools in a consistent way, and generally to make programming and computing as easy and smooth as possible.
2. The display screen will be the user's window on all activities in the local computer and on those activities within the network that concern the user. Thus the user is monitoring a set of concurrent activities, each of which is achieving some part of the user's task. These activities all use the screen to communicate with the user; those activities that are currently dormant or idle use only a small amount of screen space, while those of vital concern at the moment (e.g., a file being edited) use a larger amount. The user manages these activities by managing their presentations on the screen. For example, while a user is waiting for a long program to compile, he may want to send a message or to edit a documentation file. However, the compilation in progress must report important information (e.g., errors) to the user; such information is presented in full on the display, rather than simply a terse "compilation finished" message. Many of the programmer's tasks fall into this class: compiling, formatting documents, printing, plotting, and even running application programs. By providing for the simultaneous execution of a number of tasks, the user's productivity is enhanced.
3. We shall pay careful attention to the command language, in order to develop consistent

conventions wherever possible, to design languages that avoid confusion, and above all, to measure carefully the use (and misuse) of the commands. We believe that the basic SPICE software should set a *style of interaction* that provides a model for other designers constructing programs that run under SPICE. Such a model helps lead to consistency among command languages in the system. An example is the influence of the TENEX, or TOPS-20, command language on programs written outside the system by other individuals.

4. The user should be unaware of where resources are located, unless it is essential for him to know. The specific location of a file should be of no concern to the user; all file systems, local and remote, are transparent to the user. SPICE will be designed to encourage routine use of interprocess communication, which will appear to be the same whether or not the processes are running on the same machine.
5. SPICE will try to integrate communication modalities (text, graphics, voice) throughout the system.
6. Finally, graphics will become a way of life for the user/programmer. A substantial library of graphics software will be necessary for the SPICE environment itself, and the easy availability of this software, along with the universal availability of display screens within the environment, will influence users to make better use of graphics in their own work.

SPICE software has some key elements that will receive special attention during planning and specification. These elements will be constructed (or adapted from existing software) with considerable care:

1. **The programming environment**. A suitable high-level language will be selected for use in building SPICE and in providing a major programming environment for SPICE users. It is likely (though not certain) that Ada, the new language being designed for the Department of Defense, will be chosen as the base. The programming environment built around the selected language will include a language-oriented editor, compiler, linker, symbolic debugger, and runtime library, all integrated into the SPICE style (e.g., using the display for interaction). In addition, it is likely that tools will be developed to help manage small and medium-sized programming projects such as those used to build SPICE. These tools may include module specification standards, documentation standards, tracing and history records, etc. The programming environment will remain under active modification throughout the project. We expect significant growth in the environment as subroutine libraries are built to handle the more common programming aspects of SPICE: communication, graphics, etc.
2. **Document production tools.** The major components are a text editor, a text formatter, and graphical illustration facilities. We already have extensive experience with SCRIBE, a text formatter that carefully separates the author's intent from the tedious details of layout and typography. We shall certainly build a screen-oriented editor, based on experience with EMACS and the Alto editor.
3. **Personal communication: electronic mail.** One of the first pieces of SPICE to be built will be a multi-media message system that allows text, graphics, and voice to be transmitted and stored as messages. This system must interface smoothly to existing message systems in use in the department (e.g., ARPAnet messages).
4. **The operating system.** Operating systems for personal computers differ from those for time-shared or "virtual machine" (e.g., VM370) use in several ways. Protection, scheduling, and performance fairness decrease in complexity and importance, while facilities for interacting with the user, such as display management, input event handling, and voice I/O, become critical constraining factors. SPICE may be built initially using whatever system support is available on the hardware, and using a subroutine library to create a "SPICE virtual machine" within that operating system. Eventually, extensive modifications to the operating system or a complete

overhaul may be undertaken. If we build SPICE on top of an existing operating system, the easy modifiability of that system will be of great importance. This means that the system should be written in a high-level language, it must be modular and well documented, and it must not be too bloated with extraneous features unless these can be stripped away in a modular fashion. If the proposed system does not meet these criteria, we are better off working from scratch. If feasible, however, we will try to include the necessary support to run the manufacturer's other software products.

5. **Network support.** SPICE depends on high-performance communication software, integrated into the environment so that most communication is transparent to the user. We shall base our designs on the Xerox Pup and ARPA INTERNET/TCP efforts. It is quite likely that SPICE will support these two protocols in order to communicate with other equipment in our environment and with colleagues on the ARPAnet. The SPICE design will certainly require some new protocol designs (e.g., for transparent file access), but it is possible that lower-level protocols may be copied (e.g., use INTERNET datagrams for all communication). We have an interest in systematizing the construction of high-level protocols to remove much of the tedium and unreliability of the design and construction process.

6. **Network services: printing.** We intend to exploit the Xerox Dover printer given to the department for the bulk of our printing. We also plan to drive a photo-typesetter in the university's printing services department. The development of a uniform but flexible format for communicating with various printing devices will have a high priority.

7. **Network services: file storage.** A shared file system will be accessible over the network. The file system is one of the "central" components of the SPICE environment as it acts as a central repository for sharing documents, programs, documentation, calendars, mailboxes, handbook information, and personal and departmental records. We shall carefully study existing central file systems, including a system given to us by Xerox, before designing the SPICE central file system. Some general objectives, such as transparency, reliability, extensibility, security of confidential information, and automatic archiving, are clear. Less clear are the performance requirements, dictated in large part by the ways in which people and programs use the facility.

Because the research work of the department will be increasingly conducted within the SPICE environment, the body of useful software available in that environment will grow steadily. The development of additional facilities to meet the needs of our established research projects will *not* form part of the SPICE project proper, and consequently cannot be part of an agreement between CMU-CSD and a partner. In most cases, however, software resulting from our government-funded research is freely available in the public domain. The existence of such facilities on SPICE should greatly enhance the system's value in a variety of market areas. The following are examples of research areas outside the immediate scope of the SPICE project, but which will almost certainly make use of the SPICE environment:

- **Programming environments.** In addition to a basic environment used to construct SPICE itself, it is likely that other environments will be developed. Chief among these will be LISP, still a favorite vehicle for many researchers, because of its representation flexibility and fully interactive nature.
- **Programming methodology.** Considerable interest in the department on tools and techniques for constructing large programs with teams of people is likely to lead to prototype systems for managing software development.
- **Artificial intelligence.** The SPICE environment is an attractive one for those engaged in research in speech understanding, natural language processing, image understanding, knowledge bases, and in constructing expert consultant systems in various domains. Interest in the department in

production system architectures will probably lead to production system interpreters for SPICE.

- **VLSI design aids.** The SPICE hardware is almost ideal as a design station for the VLSI designer. The department's emerging interest in VLSI design will probably lead to design tools in the SPICE environment.

- **ZOG.** This is a highly interactive system for presenting large, complex data bases to people. This system is patterned after an experimental system used by doctors to manipulate medical records. At CMU-CSD, the project is focussed on understanding the characteristics of the man-machine interaction. ZOG is used in our environment to coordinate multi-person projects, such as the Ada compiler project. It is likely that ZOG will be applied in a similar way to the management of SPICE, and perhaps to the task of presenting SPICE documentation to its users.

The Partnership

It is intended that the design of SPICE be a joint effort in which the experience and expertise of one partner will be made available for the benefit of the other. CMU-CSD will design and produce the software and the partner will produce the hardware.

The project will involve a long-term commitment from both parties as the design is expected to evolve over a period of about 5 years. We plan to begin as soon as possible, to have an environment with intermediate capabilities within three years, and to have a complete SPICE system by 1985. We expect the hardware to be in use here until about 1990 and the software until approximately 1995. The lifetime of SPICE in the commercial marketplace should be considerably longer.

We want to emphasize that SPICE requires a long-term commitment, both from CMU-CSD and from a partner. The commitment is necessary so that CMU-CSD can mobilize design and construction talent within the department and justify devotion of the department's resources to the project. Our eagerness to undertake SPICE stems from a real desire to have it and to use it, so that our research productivity increases.

Note that the partnership we propose is limited to the *Computer Science Department* of CMU. We do not propose that the venture attempt to make the SPICE environment available through the CMU Computation Center to the university as a whole. Of course, when a SPICE environment becomes commercially available at a reasonable price, we would anticipate its use throughout CMU and at many other universities.

What the partner provides

The partner provides all of the SPICE hardware, in a series of deliveries starting as soon as possible, and spread through 1985. Initially, approximately 10 SPICE machines with the functional properties outlined above will be required, and a total of 200 will be needed by 1985. The details of a delivery schedule and configurations of the SPICE computers will be negotiated. We anticipate three phases in the effort:

1. The first machines to be delivered will be used to begin development of a programming environment, and to build a smooth bridge to existing CMU-CSD equipment.
2. Machines in the next delivery batch (approximately 25) will be used for the construction of the basic SPICE environment. These machines must provide *all functions envisioned for the SPICE hardware* so that our designs exploit the system fully. There must be sufficient numbers of machines available at this stage to accommodate the SPICE personnel.
3. In the third phase, the rest of the machines.

Any regular software products of the partner that are applicable to the SPICE environment will be made available to CMU-CSD without charge. Source files must be included. In addition, the partner may wish to make available any internal developmental tools that might facilitate the timely and robust construction of SPICE software. What constitutes appropriate software cannot be decided in advance, but it is clear that SPICE will be a cooperative venture in which the best interests of both parties will be served by maximizing the productivity of each. CMU-CSD will observe any proprietary interest in the software provided by the partner.

The partner may wish to establish some mechanism to promote the transfer of SPICE technology from

CMU-CSD into its own organization. This will probably involve the temporary assignment of personnel to CMU-CSD. We are prepared to host a modest number of the partner's employees to participate actively in the design and construction of SPICE. These people can be chosen and replaced at the partner's option, but management control for SPICE as a whole will rest with CMU-CSD (see below).

The partner will have to offer considerable financial assistance to CMU-CSD, in the form of hardware discounts or gifts, grants, etc. The partner must view the SPICE effort as a joint effort, in which CMU-CSD supports a rather large staff doing software design and construction, and the partner supplies the computing hardware. Although we anticipate with confidence that prices for SPICE hardware will eventually be reduced to the point that CMU-CSD and other similar institutions can afford personal scientific computing, we cannot afford commercial prices of such computing now.

What CMU-CSD provides

CMU-CSD will provide the manpower and expertise to design, develop, and implement SPICE, conducting the necessary investigations of experimental systems to bring one or more versions of SPICE into operation.

Given the nature of the CMU-CSD university environment and the distributed community effort involved, the exact amount of system development to go into SPICE environment cannot be specified in advance nor will it be possible to measure or account for it accurately after the fact. Moreover, the extent of CMU-CSD manpower commitment to SPICE might be the subject of negotiation with a partner. Nevertheless, we wish to illustrate the scale of commitment that CMU-CSD expects to make: *over the five years of SPICE development, more than 100 man-years of system development by high quality personnel will be provided.* The effort will probably be substantially larger. There will be significant amounts of manpower devoted in each year.

The manpower for the development of SPICE will come from many sources, reflecting the importance that CMU-CSD attaches to obtaining such a computational environment. Control and direction of the entire project will rest with the faculty, who will accept responsibility for the system. However, it is expected that everyone in the department will make contributions of varying kinds. A staff of about 10 full-time people will be devoted to SPICE development. Some fraction of the research time of every research associate will be given to the project and graduate students will also contribute significantly. Undergraduates will be used as part-time workers. SPICE will also be used as a source of projects in software engineering courses. We also expect to lay some form of "levy" in which everyone (faculty, students, staff) in the environment will be expected to make some contribution to SPICE. A rough estimate of the relative contributions of these sources in a typical year of the project is:

Source	Man-years
Full-time staff	10.0
Full-time student internships	2.0
Research Associates	1.5
Graduate students (summer + acad year)	6.7
Undergraduate part time	3.0
Undergraduate class work	?
Faculty	1.0
Levy (not counted in above)	5.0
Total	29.2

Because the department will grow from its present size of about 120 to approximately 200 during the period in question, many of the figures above may grow accordingly.

The design and development of SPICE will be under CMU-CSD's exclusive control. However, any specific wishes of the partner will be accommodated by negotiation. CMU is a university, devoted to open research and funded in part by public money. We must therefore conduct the design and development of SPICE in an open fashion, in which the partner's personnel may participate as equals. The research that CMU-CSD does is available to the scientists who perform it for their own publication. Short term confidentiality can be granted to the partner as necessary to protect his interests.

It is our goal that the central SPICE software eventually be distributed and maintained by the partner; this will insure the widest possible availability of the SPICE environment and some degree of standardization. Precedents for this sort of transfer exist in the TENEX/TOPS-20 system and the MULTICS system. Any modifications necessary to make SPICE a commercial product will be the responsibility of the partner.

Heterogeneity

Although we expect that the SPICE hardware and software efforts will be sufficiently exciting to capture much of the department's imagination, innovation, and design effort, SPICE will never be the only activity in the department. CMU-CSD is committed to a vigorous research and education program, which will not be curtailed in order to pursue SPICE. Wherever possible, we will encourage overlap of the pursuit of SPICE with research and education.

Likewise, SPICE hardware will dominate the department's facilities, but will never exclude other equipment. Other manufacturers' equipment will always have a place at CMU-CSD for several reasons:

1. Our present hardware has a useful life that overlaps the SPICE future. Some of our equipment has 5-8 years of remaining life.
2. Some computing needs of our research will require special-purpose computing. For example, we are working with CDC to design a high-speed computer for image-understanding applications. Such machines will exist only in small numbers, and often without general-purpose programming facilities.
3. Experimental hardware may be constructed, using commercially-available parts as a base. Our two multiprocessors, C.mmp and Cm*, have both used minicomputers (PDP-11's and LSI-11's

respectively) as the basic processor.

4. Gifts and grants from manufacturers or foundations may add considerable equipment to our environment. The Xerox equipment grant is a case in point: 17 Alto computers, a file server, a printing server, and Ethernet communication link have been given to the department. We expect considerable use of this equipment, and anticipate that our association with Xerox will continue.

Such heterogeneity must be tolerated by our SPICE design and, indeed, by any commercial personal computing product. The need to connect personal computers to other equipment in smooth ways is likely to be a persistent feature of this style of computing. The necessary heterogeneity in the department should provide insight into designing personal computing environments that can accommodate connections to other computing devices of all sorts.

Advantages to the partner

We summarize here the advantages that we see for the partner in entering into the SPICE venture with us.

The most important advantage is the development of a SPICE-like environment. In our view, this is analogous to the development of a sophisticated time-sharing environment around the PDP-10s by a small set of universities in the late sixties and early seventies. We believe the style of computing exhibited by SPICE will be the dominant "high technology" style of the mid-eighties. Not until the lessons of the SPICE style are learned, will the computer field be prepared to evolve the next style beyond that. Experience gained in this project will determine whether or not CMU-CSD and a partner will play significant parts in future developments. *We wish to emphasize that this proposal has to do with our relative positions in this development, not with whether such a development will occur.*

There is much to be learned about how to move from today's time-shared environments to the world of personal scientific computers, and about how best to exploit the opportunities of this new world when we get there. It may be several years before decreasing hardware costs make an environment like SPICE commercially attractive on a large scale, but it is important to begin learning about this new world now, before the economic event occurs. What CMU-CSD actually does for the partner is to blaze the trail toward this new environment: to produce the initial iteration of systems, to find out some of what works and does not work, to take some of the risks, and to move rapidly down this path.

An extremely important feature of SPICE is that it integrates the coming personal computing into a framework that is consonant with organized effort. The immediate response to the ability to have personal computers is the wholesale development of standalone micros. While there are important markets (e.g. recreational computing) where this sort of "line of least resistance" is probably appropriate, learning how to exploit the personal computer to attain more substantial computing goals is critical and difficult.

A further important advantage to the partner is that he obtains the efforts of creative first-rate computer scientists in the development of a highly marketable system, designed from the start to meet the needs of the scientific community. The partner harnesses not only our talent at implementation, but our ideas and judgement as well. Proof that such collaboration between industry and a university can be mutually beneficial may be illustrated by the advantage that has accrued to DEC from the university contributions to the PDP-10

development in the last decade. A joint SPICE venture will mobilize a community of over 100 PhD-level computer scientists and engineers (we include here senior graduate students) who will be deeply concerned with the partner's personal computing system, its problems and its future shape. This is a body of talent and expertise that would be hard for a partner to duplicate.

Finally, there are a number of benfits of a marketing kind that will come from the SPICE project, which will be open to public view. While we believe these benfits will have great value to our partner, we trust that the partner has more skill than we do in evaluating such benefits.

The venture requires that the partner bear almost all the cost of the hardware. Given the current expense of computers it might seem that CMU-CSD should bear some fraction of the cost. In fact, the total contribution we propose to make is probably more than the total cost of the hardware. More important, in terms of the success of this enterprise, the funds available to CMU-CSD to extend its computing facilities are limited. Funds that do not have to be paid for hardware will go instead towards the SPICE software effort. A significant fraction of the SPICE manpower will be supported by CMU-provided funds.

Advantages to CMU-CSD

This proposal constitutes a major commitment on the part of CMU-CSD, which carries with it high costs. We enter into it because we feel that these costs are offset by great advantages. The advantages are not entirely summed up in the total computing power of the SPICE hardware we obtain. We therefore indicate the reasons why we put forth this proposal at this time, in order that a partner may understand how we view the proposal.

Though CMU-CSD has a substantial computing environment (one large KL10, two large KA10s, two experimental multiprocessors, C.mmp and Cm*, and several experimental PDP-11/40 systems) it is currently completely saturated. Not only is it saturated, its level of sophistication has failed to keep pace with other advanced computer research organizations, especially in industry. Moreover, our computing needs continue to grow. Thus, at this time we need to do something substantial about our computing environment, both quantitatively and qualitatively.

It is clear to us that the days of the PDP-10 are now numbered. Though the PDP-10 system will no doubt remain useful for some time for limited growth and continued operation, it is no longer the obvious system of choice for extension into the future. Unfortunately, there is no good alternative to the PDP-10. The problem is not that there are no appropriate computing machines. Many powerful new engines are being developed. The problem is that of obtaining an appropriate software environment for future research.

We view with immense concern and trepidation the alternatives that face us in obtaining this new environment. In particular, there appears no way to avoid major software investments on our part, whatever course we choose. Leaving the PDP-10 world simply has its costs. Staying with it at this point saves the software costs, but spends our funds now on a path that will have to be aborted soon, when the new direction taken by the world of advanced computer research becomes manifest.

The current proposal is the best alternative that we can find to resolve this issue. It faces up to the immense

software investment that will be necessary to create a comprehensive new environment for our own work. But it makes that effort worthwhile by promising to provide an environment that will be useful through the late eighties and by offering the possibility of major contributions to the shape of the computing environment of the eighties.

CMU-CSD is much concerned about the problems of diversity, and of system and facility maintenance, both hardware and software. We do not desire to be responsible for software or hardware maintenance on idiosyncratic systems. Some of this is forced on us by the nature of research and advanced development, but it is a major source of cost and instability. The adoption of a SPICE world limits the diversity of physical systems, especially at the basic facility level. That many of the advanced software developments required in SPICE would find their way into partner-maintained systems is an immensely attractive aspect of the present proposal. We would expect SPICE to evolve continually to become less idiosyncratic and more dependent on standard software and hardware. (We see this as simply limiting the maintenance load, since new advances will continually create additional diversity and idiosyncratic maintenance problems.)

The emphasis on obtaining a large number of SPICE machines soon is not only the rational course for obtaining SPICE, it is essential if we are to commit our full energies to this course single-mindedly. We cannot overemphasize this. Gradual introduction of SPICE machines will have the effect of producing an environment that is dispersed over many types of machines with a much less focused (and much smaller) software effort. In blunt terms, if there is no assurance that we are building ourselves a major environment for tomorrow, the CSD community will not be able to convince itself collectively that it should muster the immense effort that is being proposed. This is not just a question of decisions by the senior people in CSD. In a university research community such as CMU-CSD, perceptions and commitments are arrived at and maintained independently. Thus the partner's commitment to a substantial set of initial SPICE computers constitutes a pre-emptive move that sets CMU-CSD fully on this course for the next few years.

Recent Accomplishments of CMU-CSD

SPICE is an ambitious undertaking, but we believe that it is well within the capabilities of our department. The Computer Science Department of Carnegie-Mellon University is one of the oldest and most distinguished computer science departments in the world, consistently ranking in the top three along with Stanford and MIT. We list here a number of recent accomplishments of CMU-CSD, as a rough guide to the level of ability that we will be able to bring to this task. This list makes no attempt to be exhaustive; rather, we highlight those accomplishments in areas related in some way to SPICE.

Multi-processor Systems:

- Designed a writable control store for PDP-11/40 system (the 11/40E), later used in the C.mmp system.
- Developed the first multi-mini-processor (C.mmp) with a non-trivial number of processor elements (16).
- Developed the first flexible object-oriented operating system for a multiprocessor environment (Hydra) and demonstrated its use.
- Developed a flexible, hierarchically structured architecture for multi-processor systems (Cm*) and demonstrated a prototype system with 50 processors.
- Developed an object-oriented operating system for Cm* (StarOS).
- Formulated and demonstrated language constructs for parallel processing (Algol-68 on C.mmp and Cm*).

Software Technology:

- Formulated one of the first programming languages for software development (Bliss).
- Developed a highly optimizing compiler for a system-development language (Bliss/11).
- Developed a modular software architecture for the design of a family of operating systems.
- Formulated concepts for a language (Alphard) with facilities for abstraction and verification of programs.
- Formulated the concepts of a production quality compiler-compiler.

AI, Speech, and Image Understanding Systems:

- Developed a highly efficient implementation for production systems (1977).
- Developed the first 1000 word vocabulary connected speech understanding system (Harpy, 1976) with over 90% sentence accuracy.
- Developed the first connected speech understanding system that works with male and female speakers, in terminal room environments, and capable of live spontaneous demonstration (Harpy, 1976).
- Developed the first 1000 word vocabulary speech understanding system operational on a low-cost mini-computer (LOCUST, 1977).

- Developed the first image segmentation algorithm to work on a wide variety of natural scenes.
- Developed a new instruction set in firmware useful for image applications.
- Designed an integrated image data base containing both symbolic and signal information.

Responses

This paper is not a request for formal proposals, but rather an invitation to join in a partnership to build SPICE. Companies interested in being a partner in the venture are invited to explore SPICE informally with CMU-CSD. Arrangements to discuss SPICE can be made by contacting:

Howard Wactlar
Director of Research Facilities
Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa. 15213

412-578-2571

Although initial discussions will be informal, it is likely that formal proposals will have to be prepared eventually by any companies interested in joining with CMU-CSD.

References

Although no single published document describes the SPICE environment we aspire to, most of the ideas in SPICE are borrowed from existing systems. This section attempts to point briefly to other reading.

Personal computing (although not personal *scientific* computing)

1. Alan Kay, "Microelectronics and the Personal Computer," Scientific American, Sept. 1977.
2. Alan Kay and Adele Goldberg, "Personal Dynamic Media," *Computer*, March 1977.

Personal computers

1. C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, "Alto: A Personal Computer," to appear in in D. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures, Readings and Examples*, second edition.
2. "PERQ," Three Rivers Computer Corp., 160 N. Craig St., Pittsburgh, Pa. 15213.
3. Alan Bawden, et al, "Lisp Machine Project Report," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, AIM 444, August 1977.
4. G. L. Steele, Jr., and D. A. Moon, "CADR," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, April 11, 1978 (MIT LISP machine report).

User interface through display

1. W. Teitelman, "A Display-Oriented Programmer's Assistant," *Proc. 5th Int. Joint Conf. Artificial Intelligence*, pp. 905-915, 1977. Expanded version available as Xerox Palo Alto Research Center publication CSL 77-3.
2. D. C. Swinehart, "COPILOT: A Multiple Process Approach to Interactive Programming Systems," Stanford Artificial Intelligence Lab. AIM-230, July 1974.
3. K. A. Lantz and R. F. Rashid, "VTMS: A Virtual Terminal Management System for RIG," TR 44, Computer Science Dept., Univ. of Rochester, Rochester, N.Y., May 1979.
4. G. Robertson, A. Newell, and K. Ramakrishna, "ZOG: A Man-Machine Communication Philosophy," Carnegie-Mellon Univ., Technical Report, August 1977.

Operating systems

1. B. W. Lampson and R. F. Sproull, "An Open Operating System for a Single-User Machine," 1979 Symp. on Operating System Principles.
2. D. D. Redell, et al, "Pilot: An Operating System for a Personal Computer," 1979 Symp. on Operating System Principles.

Graphics

1. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics,* second edition, McGraw-Hill, 1979.
2. See Alto paper, above.

Communication protocols and programming

1. D. R. Boggs, J. F. Shoch, E. A. Taft and R. M. Metcalfe, "Pup: An Internetwork Architecture," to appear in *IEEE Trans. Comm.*, January 1980.
2. J. Postel, "Internet Datagram Protocol - Version 4," IEN 80, USC Information Sciences Institute, February 1979.
3. J. Postel, "Transmission Control Protocol - Version 4," IEN 81, USC Information Sciences Institute, February 1979.
4. R. F. Sproull and D. Cohen, "High-level Protocols," *Proc. IEEE*, 66(11):1371-1386, November 1978.
5. J. A. Feldman, "High Level Programming for Distributed Computing," *CACM*, 22(6):353, June 1979.

Communication hardware

1. R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *CACM*, 19(7):395-404, July 1976.

Circulate & Return to me

1 ~~HF~~ 5 ~~Dick~~ Snyder

~~2 Bell D.~~ 6 Wayne Rosing

~~3 Dick~~ got back 1/19/80

4 BJ

Put in file in my office (you know where?)

PBS

APR 7 1980

OCT 9 1979

10-11

Proposal for a Joint Effort in Personal Scientific Computing

Carnegie-Mellon University

Department of Computer Science

Dick: Yep the target!, might not believe the cost & date by a few years but the target is right & we must keep VAX pressing for the target - also for the next 10 years we got to make money on stupid relatively fixed applications on low cost smart terminals with moderate comm speeds.

Snyder: I agree. Today we have TIs providing personal computers (our terminals) connected via memory. Next step is better terminals (SPICE) connected with local area networks

ROSING: THEY FORGOT ABOUT CENTRAL COMPUTER RESOURCES - THIS SYSTEM WILL REQUIRE MORE CENTRAL MIPS/USER. IF THEY PAY 50% OR SO, IT'S A POSSIBLY REASONABLE PROJECT, BUT I DON'T SEE US GETTING A LOT IN RETURN EXCEPT GOOD PRESS + A SHOWCASE. AFTER A FEW YEARS, HOWEVER, NEW APPLICATIONS SHOULD SURFACE - A PRODUCT OF THE NEW "ENVIRONMENT". SIGNIFICANT THOUGH, IS THEIR UNDERSTANDING OF THE ~~USE~~ INTEGRATION OF VOICE INTO THE NETWORK.

Proposal for a Joint Effort in Personal Scientific Computing

Carnegie-Mellon University

Department of Computer Science

August 23, 1979

This proposal is being sent to a number of computer manufacturers to solicit their interest in joining with the Carnegie-Mellon University Computer Science Department in building SPICE, a personal scientific computing environment, using hardware that will be commercially available on a large scale.

Table of Contents

Introduction

The era of time-sharing is ending. Time-sharing evolved as a way to provide users with the power of a large interactive computer system at a time when such systems were much too expensive to dedicate to a single individual. The size of these systems made a *qualitative* difference in the kinds of service they provided, and the fact that many users ran on the same machine made possible a level of user cooperation and communication that could not have been achieved on small individual machines.

Recent advances in hardware technology are opening up new possibilities. The level of capital investment which today provides each user with a small slice of a time-shared machine and a crude CRT terminal will, by the mid-1980's, provide that same user with his own powerful machine, far more powerful than today's microprocessors and equipped with such features as high-resolution color graphics and audio I/O devices. This development will enable the user to avoid many of the compromises and limitations inherent in time-sharing. New high-speed network technologies make it possible to move to this personal computing environment without foregoing the attractive features of time-sharing: shared information, good inter-user communication, and the sharing of expensive peripherals. We are entering a new world of versatile personal computers, as different from the world of time-sharing as that world was from batch-processing. It will take years of work at many different installations to learn how best to exploit these new opportunities.

We in the Computer Science Department of Carnegie-Mellon University (CMU-CSD) intend to participate in creating and exploring this new world. A start in the exploration has already been made at Xerox PARC with the ALTO environment and its progeny and at MIT with the Lisp Machine project. Both of these projects are based on specially-designed prototype hardware that will not be available commercially. In creating our own personal computing environment, CMU-CSD could follow a similar route, or we could wait for some established manufacturer to market the appropriate hardware. But we would prefer to follow a third route: we would like to work closely with some computer manufacturer to develop a personal computing environment based on a machine that will be commercially available on a large scale. We are prepared to make a substantial long-term commitment of our time, expertise, and imagination to this partnership; we are looking for a comparable commitment of resources by our partner. We propose to call the resulting environment SPICE, for Scientific Personal Integrated Computing Environment.

A cooperative arrangement has three main advantages for us. First, we avoid the logistical problems associated with designing and building our own hardware. Second, under the proposed partnership we could obtain the hardware for a full-scale SPICE environment sooner than we could otherwise afford. Third, the results of our work are immediately portable to other sites that have or could buy the same equipment. This would allow us to share more effectively programs and techniques developed within this larger community. We believe that the impact of this kind of personal computing can be revolutionary, but the revolution cannot

occur until the hardware becomes generally available through commercial channels.

We believe that our partner in this enterprise will derive great benefits as well. He will be able to draw on the proven experience of CMU-CSD to provide constructive technical advice in the development of a new product which will define the state of the art. In addition, since he will be among the first to develop a personal computing system for the commercial market, he will be uniquely placed to become a leading supplier. His product will create the market and also provide a standard against which potential competitors will be compared.

In the remainder of this paper we will describe in more detail our view of what the personal computing environment could be like, the technical requirements of the hardware and software, and the kind of partnership that we would like to form .

The Vision

Overview

We expect that CMU-CSD in the mid-1980's will be a community of about 200 researchers engaged in research and development activities in all parts of computer science: programming systems, computer architectures, VLSI design, data bases, computer networks, human-computer interaction, artificial intelligence, theory of computation, etc. This diverse community will be served by a common computing environment which will provide an ever-growing collection of software tools and capabilities. In addition to providing facilities for computer science research, the environment must also provide a number of office-automation services: report preparation, personal communication, management of meeting schedules, maintenance of departmental records (some of which are confidential), etc.

The environment we envision has the following components. (More technical details and some rationale for these requirements will be provided in a later section.)

1. **A set of personal computers**, roughly one per researcher, each with something like the power of a medium-sized time-shared computer of today. We see a need for the following features in these machines:

Speed:	10^6 macro-instructions/second.
Control store:	Room for at least 16K micro-instructions, reasonably structured and user-writable.
Virtual address:	Provision for 2^{30} to 2^{32} bytes of address space, smoothly addressable.
Primary memory:	1 Mbyte, minimum.
Secondary storage:	100 Mbytes, suitable for paging.
Display:	Color, bit-mapped, high resolution (1K x 1K pixels); with a good keyboard and a "mouse" or other pointing device.
Other:	Transducers and digital converters for audio input and output. Provision for attaching a TV camera or other device for input of static images.

   This describes the *typical* configuration; it must be possible to add additional main memory, control memory, secondary storage, special processing boxes, and peripherals as needed. The architecture must be sufficiently powerful and general to assure a long, stable lifetime, since our investment in the software will be considerable. We would expect that by the mid-1980's such systems could be priced around $10,000, though they might cost many times that figure today.

2. **A high-bandwidth network** connecting all of the processors in the community, including some experimental processors and special-purpose devices.

   In an environment such as ours, it is very important that the network be able to accommodate alien, highly specialized devices and interfaces to other networks in addition to providing efficient service to the standard SPICE machines. We expect to develop modes of operation in which larger tasks can be divided into parts and passed out to idle processors on the network. This local network will have reasonably transparent connections to comparable networks at other installations around the country, even if these sites use hardware that is different from ours.

3. **A shared file system** to store information for all users on the network.

   This file system may run on one central file computer or it may be distributed around the network, but it will appear to users as a single unified facility. This repository will promote

sharing of information within the community, and will make it possible to keep all files synchronized and up to date. Backup copies will be made automatically, and great care will be taken to ensure that this system is never unavailable for long. A large library of reference material will be available on-line using video disk or other comparable technology.

4. **Shared devices**, such as hard-copy printers, on the network.

   We would expect to have at least one high volume, high resolution xerographic printer for most output and a slower phototypesetting device for book-quality output. Other shared facilities on the network might include computer controlled wire-wrap and some large-screen displays for classroom use.

5. **The SPICE software environment**, which will be as important as the hardware.

   Among the facilities that must be provided are a comprehensive editing and text-processing system; mail, message-handling, and other forms of inter-user communication; a system for sending reminders and scheduling the user's appointments; extensive program development facilities in several high level languages (creation, debugging, analysis, management of programs, and library maintenance); an integrated graphics-oriented hardware design facility; and general-purpose tools such as calculators, symbolic mathematics packages, personal data-base systems, and so on. All of these systems should be easy to use, should have online documentation and tutorials, and should share command conventions wherever this is reasonable.

Note that the integrated, communication-rich environment described here differs in significant ways from the vision usually associated with the term "personal computing." The users of SPICE will form a community that shares programs and data structures, builds facilities for the common good, works on coordinated projects from many machines, and sometimes distributes a single computation among many machines. The substantial power of each machine assures that most of a researcher's work can be done within the responsive SPICE environment. The users will live on the SPICE machines, both for their technical needs and for the mechanics of everyday life: arranging meetings, communicating with other users, preparing documents, etc. None of this is characteristic of today's low-power microprocessor systems.

Advantages of a Personal Machine

Computer hardware becomes cheaper every year, while skilled manpower becomes ever more expensive and scarce. It is sound economics to provide computer users with personal computing facilities to increase their efficiency and productivity. We must stop worrying about idle machine cycles and start learning to use those cycles to benefit the user. Personal computing offers a number of advantages over a time-sharing system:

1. **Large, cycle-intensive programs** can be run efficiently.

   The scheduling and storage maintenance strategies on each machine can be tailored to meet the current needs of that machine's single user. This means that a large virtual memory system can be implemented with minimal contention and overhead, using the paging storage local to each machine. Any task can be given the machine's full resources, without regard for the needs of other users. There will be no more painful efforts to get around memory limitations and no more waiting for a large job to be swapped in.

2. **Accessible microstore** which allows the user to tailor the virtual machine to his particular task.

   In the Lisp Machine, for example, most of the Lisp run-time system is written in microcode, with room left over for the user's critical inner loops. By coding a few critical routines in microcode

the user can often obtain a substantial increase in the overall speed of a program. This may be worthwhile only for programs that will be used heavily, but in these cases it can be extremely important.

3. **High-resolution graphics** of the kind we envision will provide the user with an environment that is qualitatively different from that provided by today's standard CRT terminals.

   The added resolution makes it reasonable to divide the screen into distinct areas, each monitoring a different task. Text to be output can be displayed in exactly the form it will take on the printed page. Graphical and half-tone material can be intermixed with text, and changes in the display can be made rapidly. Color is essential for multi-layer VLSI design and should prove to be quite useful in other contexts as well. Although the CPU is not involved in maintaining static images on the screen, it must be possible to obtain a substantial burst of computation instantly, whenever the display is to be changed. Instant attention is also required if the system is to track a mouse or other pointing device properly. A time-sharing system cannot provide this level of service to all of its users.

4. **Natural communication** will be possible with the audio I/O facilities.

   A microphone, A/D and D/A converters, and a speaker will make it possible to send voice messages, to store these messages in digitized form, to annotate documents with voice, and to enter into voice dialogs with colleagues elsewhere on the network. In addition, the experience gained in the successful speech-understanding project at CMU leads us to believe that it will soon be feasible to use simple voice commands as a flexible and natural way to control programs. The necessary audio devices are now inexpensive and having them available on all machines will help us to evaluate their true potential. As with the graphics displays, these devices require occasional real-time bursts of intensive computing which would be hard to provide on a shared machine.

5. **Full availability** of all the machine's resources, independent of what other users are doing.

   No longer are big jobs forced to run at 4 a.m. No longer do simple edits become intolerably slow in the afternoon. Each user will get the same excellent level of service. Only the shared facilities and perhaps the pool of supplemental computing resources are affected by the overall system load.

6. **Reliability** is an inherent part of the environment.

   A failure in any one machine cannot bring down the others. The critical demonstration or production run can simply be moved to a working machine. The only critical resources are the central file system and perhaps the printer. These must be designed for extreme reliability or backups must be available.

7. **Extending the environment** can be done in smaller increments than in a comparable time-shared system.

   If a few new users join an installation, a few new SPICE machines can be bought for them. There will be no more agonizing decisions between running with one overloaded machine or two underutilized ones. The environment can be closely tailored to the community's needs. An important corollary of this is that an installation with only a few users can obtain essentially the same computing environment for these people that the larger institutions provide for their hundreds of users.

Some Scenarios

It is hard to describe in abstract terms how one computing environment differs from another. Many subtle differences can only be appreciated when they have been experienced first-hand. Since we are unable to convey such experience on paper, we will have to describe the feel of the SPICE world by way of some imaginary scenarios.

Scenario 1: Document Production

A user is writing a technical report at his machine. With a truly responsive text editor, it is easier for most users to compose their text on-line than to use more conventional means. The resulting prose tends to be of higher quality, since it takes much less effort to rearrange and alter text on the screen than to erase or cut and paste. As the user works, the machine can call up dictionary and other reference information and show him as he types how the finished text will look. The computer can check spelling and certain simple grammatical rules and alert the writer to problems. Illustrations and graphs can be created along the way or can be called up from the archives. To add an illustration to his document, the user may scan in an existing illustration and "paste" it into the text. When the text is completed it can be sent at once to the central repository, from which anyone on the net (or other connected nets) can access it. It can also go to the photo-typesetter for creation of hard copy.

Scenario 2: Program Development

A programmer is creating his program online. The editing program knows a good deal about the syntax of the language at hand and prevents non-syntactic constructions (unbalanced parentheses, for example) from being entered or indicates such dubious constructs with color-coding. As the program is typed in, it is automatically pretty-printed so that it always appears in the form most understandable to the programmer. Manuals, project specifications, instructions for using the editor, and an extensive library of algorithms, programs, and subroutines are all just a few keystrokes away. As a module is completed, it is passed to a project management system which coordinates the versions and specifications for each module and which remembers what tasks have yet to be done.

Scenario 3: Program Debugging

Now the program is being debugged. As the user single-steps the program, the corresponding source code is visible in one portion of his screen, with the statement being executed highlighted in color. The current stack of procedure calls appears in a second window (in a format corresponding to the source code) and the values of a dozen or so variables are being continuously displayed in a third window. As the source of a problem is discovered, it can be corrected in the source file, and the fix is reflected at once in the running image. If the external behavior of a module must be altered, an indexing system can tell the user what other modules are made inconsistent by the change. At any time, a user can get a list of inconsistencies yet to be resolved. Environments close to this have been built around the LISP language; we hope to develop comparable environments for algebraic languages such as Ada and for systems whose modules are written in several languages.

Scenario 4: VLSI Design

A logic designer is working on a new VLSI chip. As he draws the layout on his color screen, using a mouse and menu system, a constraint checking program is monitoring his work, predicting performance, adjusting the size of traces, and so on. This computation may be running entirely on the local machine or parts of it may be sent off to run on unused processors or other computing resources in the network. As he works, the designer is able to call up cookbook design elements from the on-line reference library and, if these meet his needs, plug them into the current design at desired locations. As this occurs, the constraint checker adjusts the parameters of the new element to fit the given context. When the designer is satisfied, he runs a final simulation, then transmits the design to the automated mask-making and chip-fabrication facility. A very similar scenario could be written for designing bridges, buildings, aircraft, or whatever.

Scenario 5: Multi-media Communication

Several researchers are cooperating on a large project. Each researcher is working in a different building on his own part of the project and is using his own machine. From time to time the researchers need to confer, in pairs or all together. Any user can initiate a linkup, and the others can accept or reject it. Linked users can talk to one another using their terminals' acoustic I/O and compressed digital transmission. Alternatively, they can type or sketch on their linked-together screens. If something important is said or sketched, a user can save it by sending it to his personal file of memos. When the link is terminated, the users can restore their screens and proceed with what they were doing; there is no loss of context. The same mixture of speech, text, and sketches can be sent to another user (or to everyone on a mailing list) as mail. They will see it the next time they log in or check their mailbox files.

Scenario 6: Non-Intrusive Communication

A faculty member is working on a paper, rushing to meet a deadline. He does not want to be interrupted by any but the most critical messages. Some messages must be let through, however: an expected call from a major funding agency, queries from a student preparing a companion paper, calls from the professor's spouse. The list of people and topics to be admitted is stored on the machine and compared against each incoming message. Qualifying messages are passed through at once; others are stored as mail, with the senders being notified of the situation. Optionally, some senders and topics can be ignored altogether. The topics are determined by a hierarchical scheme based on categories of keywords -- an exact match is not required. When the user finally finishes his paper, he can respond to the queued requests by mail, by linking terminals if the sender is logged in anywhere in the country, or by phone. The computer does the searching, linking, and phone dialing. All of this could, of course, be done by a competent secretary, but good secretaries are expensive, scarce, and usually work only during normal office hours.

Scenario 7: A Telephone Answering System

A researcher is working after hours at a machine far from his office. An important phone call arrives. After three rings, the phone is answered by one of the SPICE machines which is acting as telephone operator. A recorded message asks the caller to state the name of the person being sought. A speech recognition program analyzes the spoken name and locates it on a list of the system's users. The recipient is located, and the call is patched through to his terminal. If the user is not available or declines to take the call, the operator machine allows the caller to dictate a message that will be saved in audio form (compressed digital encoding). This will be sent to the intended recipient as mail. The kind of speech recognition required for identifying names from a list of a few hundred is within our current capabilities, though it is computationally expensive on conventional machines.

Scenario 8: Information Distribution

A new member has joined the department. When he first logs in to the system, the newcomer is asked for a variety of biographical information. He also has his picture taken by a television camera tied to one of the machines. The next day a message introducing the new person and displaying his picture appears among the system mail of all users. This information also goes to the file system. Whenever anyone wonders who John Smith is, the picture and biographical information can be instantly displayed on the screen of any SPICE machine.

The Design

The following is an outline of the design, which will become more specific after consultation with the partner.

A great deal of the background for the design comes from the example set by the Xerox Alto and its related software. CMU-CSD is extremely fortunate to have been given, in a generous grant from Xerox, a number of these prototype systems: several Altos, a central file system, and a Dover raster-scanned printer. Our design for the SPICE hardware and software borrows heavily from the experience of the Xerox experiment.

Almost all of the components of SPICE have been previously tried in some form; we are proposing a unique combination, but we are not venturing into completely unknown areas.

Hardware

We know of no computer presently available commercially that exactly meets the needs of SPICE, though some predecessors exist such as the MIT Lisp Machine and the Xerox Alto. Moreover, at least one new design has many of the characteristics we seek: the PERQ system designed by the Three Rivers Computer Corporation.

To reiterate the requirements listed earlier, a SPICE machine available in 1985 should have roughly the following characteristics and features:

1 MIPS execution rate
User-writable microstore
Large virtual address space
1 Mbyte primary memory
100 Mbytes secondary memory

High quality color raster display
Keyboard and pointing device

Audio input-output
Provision for connecting image input
High bandwidth network connection
External bus(es) for auxiliary devices

Packaged for office or laboratory use
$10,000 price (mid '80's)

These numbers are intended to be indicative of scale and power, not of precise features. Each of these objectives receives further comment in the paragraphs that follow.

Instruction Execution and Microcode

One of the main objectives of the basic SPICE processor must be to offer long life to SPICE software, since the investment in this software will be large. In considering this problem, we may assume that SPICE will be written almost exclusively in a high-level language.

The traditional approach to this problem is to use a standard, fixed instruction-set architecture that can be sensibly implemented with a range of performances over a number of years. For example, the IBM 360/370 architecture has this property. At the other extreme, the 8080 design was technology-induced, and is already obsolete. Because the SPICE computer will be programmed using one or more high level languages, minute details of the machine's instruction set are not terribly critical. Generally, it should allow straightforward compilation of moderately efficient code for languages such as PASCAL or Ada.

We prefer a design that harnesses user-writable microcode for the purpose of increasing software life. Several benefits accrue from the ability to write new microcode and thus modify the virtual machine seen by the programmer:

1. A powerful macro-instruction set can be implemented cheaply and in such a way that changes to the underlying technology can be made invisible to the software. This insulates the instruction set from pressures due to short-term technological constraints of the sort that have led to the grotesque instruction sets of many current microprocessors.
2. Minor extensions and modifications to the macro-instruction set can be permitted as new needs are perceived. This adds to the useful lifetime of the instruction set.
3. Customized virtual machines can be created which are specialized for different languages or tasks (e.g. to create a virtual LISP or PASCAL machine). This approach increases the efficiency of interpreted languages (relative to interpreters written in macro-code), and relieves a compiler from generating code for a machine ill-suited for the language.
4. Small "kernel" or inner-loop programs can be implemented in a very efficient way. Placing just a few critical routines in the microcode can make a dramatic difference in the execution time of some programs. For example, the operations to manipulate data in the frame-buffer display are obvious candidates for this treatment.

We feel that benefits (2)-(4) are sufficiently important that the presence of a sizable user-writable control store is essential to the success of SPICE. All uses of the microcode require that the micromachine be designed with programming in mind, and that it have a reasonably clean and complete structure. Using microcode for "kernels" requires convenient ways to invoke the kernels from macro-instructions (e.g., using unimplemented operations or traps). Using the microcode to implement language-dependent virtual machines requires several aids in the micromachine (e.g., instruction byte fetching for a byte-coded instruction set such as PASCAL PCODE). Note that the use of microcode to ease migration to new hardware, to build kernel loops, and to build specialized virtual machines is rather different from the use envisioned by most computer manufacturers, to reduce the implementation cost of a particular (fixed) instruction set.

Performance

The performance objective of 1 MIPS is, of course, a vague one. We have in mind a "typical" PASCAL statement such as **c := a+b** compiling into two instructions, each taking 1 microsecond or less.

It would be a mistake to concentrate entirely on the user interface made possible by SPICE; at least as important to us, and to the future of personal computing, is the potential of personal machines to increase the effective computing power available to each user. SPICE must be an effective computing environment for

advanced computer research and computer-aided design, not merely a collection of smart terminals.

Virtual address and memory size

Our view of personal scientific computing is not limited to small, well-bounded applications. The size of programs written by designers and researchers is large and growing. The size of a programming environment that comfortably supports a researcher is large; in addition, the SPICE vision anticipates a qualitative and quantitative increase in the amount of information summoned to aid the researcher: manuals, electronic mail, working documents, etc.

The virtual address space must be large. CMU-CSD has long ago outgrown 16-bit address spaces (PDP-11's) and 18-bit address spaces (PDP-10's). Our struggles with small addresses have slowed work in multiprocessors, operating systems, image understanding, symbolic computing, and artificial intelligence. We believe a move to a 24-bit address space is merely a stopgap (especially if the unit addressed is a byte rather than a word), but that 30 or 32 bits should be sufficient for a long time. A recent ARPA-sponsored workshop on facilities for symbolic computing seemed to achieve consensus on this point.

Sizes of primary and secondary memory must be reasonably large to match a large virtual memory:

virtual address (2^{30} bytes)	10^9 bytes
secondary memory	10^8 bytes
primary memory	10^6 bytes

Once again, these numbers are for the typical SPICE machine. The hardware must allow for some machines to have substantially more primary and secondary memory than these minimum figures suggest. We believe that addressing down to the byte level is desirable. Greater precision in the numbers in the table is pointless in the absence of a specific processor design.

Display

The raster display is a mandatory part of the SPICE computer; display output is used extensively by all SPICE software, not just by one or two "graphics programs." We believe the display's essential features are:

1. **High resolution.** To approximate a display as complex as the information on a typical 8 1/2 x 11 inch page, something like 600 x 800 bi-level (black or white) pixels are needed; more are desirable. The CPT 60 Hz monitor is adequate in terms of resolution. (To some extent, spatial resolution can be exchanged for gray-scale resolution.)
2. **Frame buffer (bitmap) representation.** The frame buffer is the only image representation technique that is completely insensitive to image complexity. Arbitrary images can be displayed by setting the intensity of each pixel in the frame buffer.
3. **Cursor.** Some provision needs to be made for a cursor. A good solution is a small bitmap buffer that can be read and written by a program just like the main frame buffer. In addition, the (x,y) position of the image of the small bitmap can be changed easily.
4. **Image-generation aids.** Some operations on the frame buffer are sufficiently tedious and simple that they deserve hardware and/or microcode assist. Examples are vector generation, character generation, and RasterOp (See Newman and Sproull, *Principles of Interactive Computer Graphics*,

2nd Edition, p. 264). We believe that a fast RasterOp implementation is especially important.

5. **Access to the frame buffer.** It is essential that an application program have direct access to the frame buffer, i.e., that pixel values can be read and written by ordinary memory references. This facility is needed because hardware image-generation aids cannot anticipate *all* uses of the frame buffer. With direct memory access, an application program can achieve any effect desired, although perhaps slowly. A corollary of this is that RasterOp must have access to the application program's memory to copy bitmaps in and out of the frame buffer and to find font representations. Generally, this requirement means that the underlying SPICE computer must have substantial memory bandwidths, so that display refreshing and instruction execution can both proceed at high speed.

6. **Color.** The SPICE computer must support color displays, perhaps as an option. A modest approach is to describe each pixel with 4 bits and to use a map to convert these values into 16 arbitrary colors for output. More elaborate color displays (up to 30 bits/pixel) will have significant application in imaging work, and might also be considered as an option.

7. **Provision for video input.** Some applications, such as interactive conferencing and document processing, can make very effective use of television or facsimile input. This facility is most easily combined with the frame-buffer display: video for a single frame is sampled and stored in the frame buffer, which can then be read by a program at modest rates. This technique will not permit the recording of real-time moving full-frame images because there is not likely to be sufficient processor or disk bandwidth to save the contents of the frame buffer before another frame must be stored.

8. **Expansion.** It is desirable to have the optional features easily added by plugging modules into a standard cage. Thus, the video input and color options will not require wholly different machines, but will be additions to the standard SPICE machine. Similarly, it would be desirable if a single SPICE computer could support additional displays, though perhaps with lower performance on the additional screens. The key point here is that the scheme for interfacing the processor to the display must provide for such extensions in a smooth and coherent manner.

Pointing Device

The requirements for a pointing device are difficult to state. It should be comfortable and easy to use. It is entirely independent of the display; it merely reports a two-dimensional position or incremental motion to the processor. Its chief uses will be for pointing to menu items shown on the screen, and for selecting text characters or graphical objects already displayed. For these purposes, it should have some buttons (on the device itself, not on the console) so that a user can indicate "this is the point I mean." Precision and repeatability are not particularly important, but reliability is essential. Something like the SRI mouse is reasonable.

Keyboard

The primary requirements for the keyboard are that it feel good and that it *be reliable*. We find the cheap keyboards on most current character terminals not only unpleasant, but destructive: they unreliably transduce the user's finger strokes. It should be possible for the processor to sense, at any instant, exactly which keys are depressed on the keyboard. The keyboard might report to the computer a bit-vector that marks depressed keys as 1 and idle keys as 0. Most keyboards do not have this property; instead, they "encode" the downward keystrokes into ASCII, modifying the encoding based on a few shift keys. Such keyboards are too

These considerations, coupled with some knowledge of what is currently feasible, lead us to choose a bandwidth of about 10 Mbits/sec. Proper design of the network software will allow SPICE to accommodate higher performance transmission systems later on. Although the software network protocols must be standardized throughout the life of SPICE (else face massive compatibility or re-programming problems), the transmission system can evolve steadily, using "gateways" to previous, slower, transmission systems.

Because the network provides access to essential services, its availability must be extremely high. A low transmission error rate is not essential (though desirable, of course) because we intend to use packet protocols to detect errors and to retransmit packets found to be in error.

External connections

Researchers often want to augment their personal machine in some way to address specific problems. In addition, special I/O gear may be added to a SPICE machine to offer a service accessible to the community via the network (e.g., printers, raster input scanners, film recorders, archival memories). We envision three kinds of external connections:

1. Simple, low bandwidth (e.g., RS 232, IEEE 488, and/or parallel "switch and relay" register), used for telephone-line connections, for small devices such as plotters and impact printers, and for connections to experimental hardware (the parallel interface is especially useful for this last).
2. Addressable, high bandwidth. This bus offers the same addressability to I/O devices that is available to user programs. This allows I/O devices to use DMA for simple purposes (like a channel) or to interpret rather complex data and control structures built by the user program.
3. Processor extensions (optional). Occasionally, an experiment may require specialized computation in sufficient quantity that it is best provided by a minor processor extension. An example is an image compressor/decompressor that transforms one description of an image stored in memory to another one, also stored in memory. Another example might be special hardware added to increase the speed of floating-point operations on those machines that use floating-point a great deal.

Package and Price

Ultimately, the SPICE computer must be packaged for convenient placement in an office or laboratory, with as much mobility as possible. We hope for a mid-80's price of $10,000.

It is likely that neither objective can be met immediately. Interim packaging solutions are necessary, although the user I/O gear (display, keyboard, pointing device, audio) should be packaged for convenient desktop use from the start.

Evolution

If SPICE is to have a long life, we must make some plan for natural evolution of the hardware as technology changes.

We believe the most important consideration is that SPICE hardware must support, from the start, *all those*

functions required in the final version of the system: audio, raster graphics, one machine per person, and substantial computing power. An interim machine that omits, for example, audio facilities, will not induce software designs that exploit the voice medium, no matter how hard we try to plan for its eventual inclusion. if the early hardware has audio input/output features, then all facets of the software environment, from message systems to programming environments, will be designed to include voice in natural ways. By similar argument, an interim machine that does not offer substantial computing power will not become the basis for significant research computing, and a SPICE scientific computing environment will simply not grow in the barren machine.

The size of the address space is a surprisingly fixed part of the SPICE hardware definition. Although it is possible in some machines to extend the address space without incurring major costs, most previous efforts to increase the addressing capabilities of a computer have been accompanied by significant re-programming efforts. A splendid example is the PDP-10, which started with an 18-bit address, and has been extended to 23 bits. The programming environments that make the PDP-10 so valuable (e.g., Interlisp) must be almost entirely rewritten to take advantage of the larger addresses. Since it is easier to build the computer with extended addressing than to adapt the software, it is, therefore, desirable to build the computer with extended addressing from the start.

The choice of an address space of about 30 bits is a compromise between feasibility and caution. Caution suggests that address space requirements grow by one bit every two years, with the current value at about 24 bits. SPICE's value in the research world will thus be limited to about 12 years. Choosing larger addresses, a more conservative approach, tends to require large word sizes as well, leading to performance and cost problems. Clearly, any proposal for SPICE hardware that accommodates virtual addresses larger than 30 bits will be well received.

It is possible to begin SPICE development with a smaller virtual address space than will be needed ultimately, but only if the transition to the ultimate large size has been very carefully planned at the outset and all software is written so that the transition is invisible. We will need something like 100 Mbytes of virtual memory even for the first machines.

The remaining facilities of the machine can be easily increased as technology improves or costs come down. More primary memory, more local storage, more control memory, and faster instruction execution are all desirable. However, any additional facilities must not introduce the need for reprogramming.

To prevent reprogramming costs, some aspects of the hardware must be fixed, even though technological advances will surely allow improvements. The fixed elements are usually termed *architectural* features, and must be supported throughout the life of a family of machines. The instruction set is probably the least critical part of the architecture, because the exclusive use of a high-level language in writing SPICE will allow conversion to a new instruction set simply (!) by modifying the compiler. The architecture of input/output devices is more critical: program structures will depend in detail on how the display is controlled, for example. Surprisingly, even the gray-scale resolution of the display may have profound impact on the software: techniques that suffice for 1-bit-per-point displays are wholly inappropriate for displays that have several gray levels. (Of course, one may achieve "backward compatibility" by operating the multilevel display as if it used only 1 bit per point, but this is presumably undesirable.)

Amendments

The hardware image we have presented is by no means inviolate. We anticipate that our partner in the SPICE venture may wish to shape the design to accommodate other product needs. For example, it is possible that some people may view a SPICE computer as a very capable graphics terminal rather than as a personal computer. The design might make certain features optional (e.g., audio) even though all machines supporting SPICE in our environment would require the feature. We hope that suggestions from the partner will help improve the SPICE hardware capability.

Software

The basic facilities which will be provided by the software are as follows:

1. Scientific (research) programming
2. Document production
3. Communication

Each of these areas covers a wide range of facilities. Programming facilities include languages, runtime systems, debugging facilities, program-construction aids, etc. Document production facilities provide for every type of text, from an informal mail message, through program texts, technical reports, and letters, to manuscripts and books. Facilities will also be included for putting graphical illustrations and voice annotations into documents. Communication is perhaps the broadest category: exchanging mail with colleagues, maintaining calendars, using voice and graphics in messages.

The SPICE system will be driven by a set of objectives that sets it apart from other personal computing endeavors (notably that of Xerox PARC). Some of these objectives are set forth below:

1. SPICE is, first and foremost, a *computing environment*. The SPICE hardware architecture, with its large virtual address space, makes the single biggest contribution to this objective. SPICE will build on this to provide interactive access to programming tools in a consistent way, and generally to make programming and computing as easy and smooth as possible.
2. The display screen will be the user's window on all activities in the local computer and on those activities within the network that concern the user. Thus the user is monitoring a set of concurrent activities, each of which is achieving some part of the user's task. These activities all use the screen to communicate with the user; those activities that are currently dormant or idle use only a small amount of screen space, while those of vital concern at the moment (e.g., a file being edited) use a larger amount. The user manages these activities by managing their presentations on the screen. For example, while a user is waiting for a long program to compile, he may want to send a message or to edit a documentation file. However, the compilation in progress must report important information (e.g., errors) to the user; such information is presented in full on the display, rather than simply a terse "compilation finished" message. Many of the programmer's tasks fall into this class: compiling, formatting documents, printing, plotting, and even running application programs. By providing for the simultaneous execution of a number of tasks, the user's productivity is enhanced.
3. We shall pay careful attention to the command language, in order to develop consistent

conventions wherever possible, to design languages that avoid confusion, and above all, to measure carefully the use (and misuse) of the commands. We believe that the basic SPICE software should set a *style of interaction* that provides a model for other designers constructing programs that run under SPICE. Such a model helps lead to consistency among command languages in the system. An example is the influence of the TENEX, or TOPS-20, command language on programs written outside the system by other individuals.

4. The user should be unaware of where resources are located, unless it is essential for him to know. The specific location of a file should be of no concern to the user; all file systems, local and remote, are transparent to the user. SPICE will be designed to encourage routine use of interprocess communication, which will appear to be the same whether or not the processes are running on the same machine.
5. SPICE will try to integrate communication modalities (text, graphics, voice) throughout the system.
6. Finally, graphics will become a way of life for the user/programmer. A substantial library of graphics software will be necessary for the SPICE environment itself, and the easy availability of this software, along with the universal availability of display screens within the environment, will influence users to make better use of graphics in their own work.

SPICE software has some key elements that will receive special attention during planning and specification. These elements will be constructed (or adapted from existing software) with considerable care:

1. **The programming environment**. A suitable high-level language will be selected for use in building SPICE and in providing a major programming environment for SPICE users. It is likely (though not certain) that Ada, the new language being designed for the Department of Defense, will be chosen as the base. The programming environment built around the selected language will include a language-oriented editor, compiler, linker, symbolic debugger, and runtime library, all integrated into the SPICE style (e.g., using the display for interaction). In addition, it is likely that tools will be developed to help manage small and medium-sized programming projects such as those used to build SPICE. These tools may include module specification standards, documentation standards, tracing and history records, etc. The programming environment will remain under active modification throughout the project. We expect significant growth in the environment as subroutine libraries are built to handle the more common programming aspects of SPICE: communication, graphics, etc.
2. **Document production tools.** The major components are a text editor, a text formatter, and graphical illustration facilities. We already have extensive experience with SCRIBE, a text formatter that carefully separates the author's intent from the tedious details of layout and typography. We shall certainly build a screen-oriented editor, based on experience with EMACS and the Alto editor.
3. **Personal communication: electronic mail.** One of the first pieces of SPICE to be built will be a multi-media message system that allows text, graphics, and voice to be transmitted and stored as messages. This system must interface smoothly to existing message systems in use in the department (e.g., ARPAnet messages).
4. **The operating system.** Operating systems for personal computers differ from those for time-shared or "virtual machine" (e.g., VM370) use in several ways. Protection, scheduling, and performance fairness decrease in complexity and importance, while facilities for interacting with the user, such as display management, input event handling, and voice I/O, become critical constraining factors. SPICE may be built initially using whatever system support is available on the hardware, and using a subroutine library to create a "SPICE virtual machine" within that operating system. Eventually, extensive modifications to the operating system or a complete

overhaul may be undertaken. If we build SPICE on top of an existing operating system, the easy modifiability of that system will be of great importance. This means that the system should be written in a high-level language, it must be modular and well documented, and it must not be too bloated with extraneous features unless these can be stripped away in a modular fashion. If the proposed system does not meet these criteria, we are better off working from scratch. If feasible, however, we will try to include the necessary support to run the manufacturer's other software products.

5. **Network support.** SPICE depends on high-performance communication software, integrated into the environment so that most communication is transparent to the user. We shall base our designs on the Xerox Pup and ARPA INTERNET/TCP efforts. It is quite likely that SPICE will support these two protocols in order to communicate with other equipment in our environment and with colleagues on the ARPAnet. The SPICE design will certainly require some new protocol designs (e.g., for transparent file access), but it is possible that lower-level protocols may be copied (e.g., use INTERNET datagrams for all communication). We have an interest in systematizing the construction of high-level protocols to remove much of the tedium and unreliability of the design and construction process.

6. **Network services: printing.** We intend to exploit the Xerox Dover printer given to the department for the bulk of our printing. We also plan to drive a photo-typesetter in the university's printing services department. The development of a uniform but flexible format for communicating with various printing devices will have a high priority.

7. **Network services: file storage.** A shared file system will be accessible over the network. The file system is one of the "central" components of the SPICE environment as it acts as a central repository for sharing documents, programs, documentation, calendars, mailboxes, handbook information, and personal and departmental records. We shall carefully study existing central file systems, including a system given to us by Xerox, before designing the SPICE central file system. Some general objectives, such as transparency, reliability, extensibility, security of confidential information, and automatic archiving, are clear. Less clear are the performance requirements, dictated in large part by the ways in which people and programs use the facility.

Because the research work of the department will be increasingly conducted within the SPICE environment, the body of useful software available in that environment will grow steadily. The development of additional facilities to meet the needs of our established research projects will *not* form part of the SPICE project proper, and consequently cannot be part of an agreement between CMU-CSD and a partner. In most cases, however, software resulting from our government-funded research is freely available in the public domain. The existence of such facilities on SPICE should greatly enhance the system's value in a variety of market areas. The following are examples of research areas outside the immediate scope of the SPICE project, but which will almost certainly make use of the SPICE environment:

- **Programming environments.** In addition to a basic environment used to construct SPICE itself, it is likely that other environments will be developed. Chief among these will be LISP, still a favorite vehicle for many researchers, because of its representation flexibility and fully interactive nature.
- **Programming methodology.** Considerable interest in the department on tools and techniques for constructing large programs with teams of people is likely to lead to prototype systems for managing software development.
- **Artificial intelligence.** The SPICE environment is an attractive one for those engaged in research in speech understanding, natural language processing, image understanding, knowledge bases, and in constructing expert consultant systems in various domains. Interest in the department in

production system architectures will probably lead to production system interpreters for SPICE.

- **VLSI design aids.** The SPICE hardware is almost ideal as a design station for the VLSI designer. The department's emerging interest in VLSI design will probably lead to design tools in the SPICE environment.

- **ZOG.** This is a highly interactive system for presenting large, complex data bases to people. This system is patterned after an experimental system used by doctors to manipulate medical records. At CMU-CSD, the project is focussed on understanding the characteristics of the man-machine interaction. ZOG is used in our environment to coordinate multi-person projects, such as the Ada compiler project. It is likely that ZOG will be applied in a similar way to the management of SPICE, and perhaps to the task of presenting SPICE documentation to its users.

The Partnership

It is intended that the design of SPICE be a joint effort in which the experience and expertise of one partner will be made available for the benefit of the other. CMU-CSD will design and produce the software and the partner will produce the hardware.

The project will involve a long-term commitment from both parties as the design is expected to evolve over a period of about 5 years. We plan to begin as soon as possible, to have an environment with intermediate capabilities within three years, and to have a complete SPICE system by 1985. We expect the hardware to be in use here until about 1990 and the software until approximately 1995. The lifetime of SPICE in the commercial marketplace should be considerably longer.

We want to emphasize that SPICE requires a long-term commitment, both from CMU-CSD and from a partner. The commitment is necessary so that CMU-CSD can mobilize design and construction talent within the department and justify devotion of the department's resources to the project. Our eagerness to undertake SPICE stems from a real desire to have it and to use it, so that our research productivity increases.

Note that the partnership we propose is limited to the *Computer Science Department* of CMU. We do not propose that the venture attempt to make the SPICE environment available through the CMU Computation Center to the university as a whole. Of course, when a SPICE environment becomes commercially available at a reasonable price, we would anticipate its use throughout CMU and at many other universities.

What the partner provides

The partner provides all of the SPICE hardware, in a series of deliveries starting as soon as possible, and spread through 1985. Initially, approximately 10 SPICE machines with the functional properties outlined above will be required, and a total of 200 will be needed by 1985. The details of a delivery schedule and configurations of the SPICE computers will be negotiated. We anticipate three phases in the effort:

1. The first machines to be delivered will be used to begin development of a programming environment, and to build a smooth bridge to existing CMU-CSD equipment.
2. Machines in the next delivery batch (approximately 25) will be used for the construction of the basic SPICE environment. These machines must provide *all functions envisioned for the SPICE hardware* so that our designs exploit the system fully. There must be sufficient numbers of machines available at this stage to accommodate the SPICE personnel.
3. In the third phase, the rest of the machines.

Any regular software products of the partner that are applicable to the SPICE environment will be made available to CMU-CSD without charge. Source files must be included. In addition, the partner may wish to make available any internal developmental tools that might facilitate the timely and robust construction of SPICE software. What constitutes appropriate software cannot be decided in advance, but it is clear that SPICE will be a cooperative venture in which the best interests of both parties will be served by maximizing the productivity of each. CMU-CSD will observe any proprietary interest in the software provided by the partner.

The partner may wish to establish some mechanism to promote the transfer of SPICE technology from

CMU-CSD into its own organization. This will probably involve the temporary assignment of personnel to CMU-CSD. We are prepared to host a modest number of the partner's employees to participate actively in the design and construction of SPICE. These people can be chosen and replaced at the partner's option, but management control for SPICE as a whole will rest with CMU-CSD (see below).

The partner will have to offer considerable financial assistance to CMU-CSD, in the form of hardware discounts or gifts, grants, etc. The partner must view the SPICE effort as a joint effort, in which CMU-CSD supports a rather large staff doing software design and construction, and the partner supplies the computing hardware. Although we anticipate with confidence that prices for SPICE hardware will eventually be reduced to the point that CMU-CSD and other similar institutions can afford personal scientific computing, we cannot afford commercial prices of such computing now.

What CMU-CSD provides

CMU-CSD will provide the manpower and expertise to design, develop, and implement SPICE, conducting the necessary investigations of experimental systems to bring one or more versions of SPICE into operation.

Given the nature of the CMU-CSD university environment and the distributed community effort involved, the exact amount of system development to go into SPICE environment cannot be specified in advance nor will it be possible to measure or account for it accurately after the fact. Moreover, the extent of CMU-CSD manpower commitment to SPICE might be the subject of negotiation with a partner. Nevertheless, we wish to illustrate the scale of commitment that CMU-CSD expects to make: *over the five years of SPICE development, more than 100 man-years of system development by high quality personnel will be provided.* The effort will probably be substantially larger. There will be significant amounts of manpower devoted in each year.

The manpower for the development of SPICE will come from many sources, reflecting the importance that CMU-CSD attaches to obtaining such a computational environment. Control and direction of the entire project will rest with the faculty, who will accept responsibility for the system. However, it is expected that everyone in the department will make contributions of varying kinds. A staff of about 10 full-time people will be devoted to SPICE development. Some fraction of the research time of every research associate will be given to the project and graduate students will also contribute significantly. Undergraduates will be used as part-time workers. SPICE will also be used as a source of projects in software engineering courses. We also expect to lay some form of "levy" in which everyone (faculty, students, staff) in the environment will be expected to make some contribution to SPICE. A rough estimate of the relative contributions of these sources in a typical year of the project is:

Source	Man-years
Full-time staff	10.0
Full-time student internships	2.0
Research Associates	1.5
Graduate students (summer + acad year)	6.7
Undergraduate part time	3.0
Undergraduate class work	?
Faculty	1.0
Levy (not counted in above)	5.0
Total	29.2

Because the department will grow from its present size of about 120 to approximately 200 during the period in question, many of the figures above may grow accordingly.

The design and development of SPICE will be under CMU-CSD's exclusive control. However, any specific wishes of the partner will be accommodated by negotiation. CMU is a university, devoted to open research and funded in part by public money. We must therefore conduct the design and development of SPICE in an open fashion, in which the partner's personnel may participate as equals. The research that CMU-CSD does is available to the scientists who perform it for their own publication. Short term confidentiality can be granted to the partner as necessary to protect his interests.

It is our goal that the central SPICE software eventually be distributed and maintained by the partner; this will insure the widest possible availability of the SPICE environment and some degree of standardization. Precedents for this sort of transfer exist in the TENEX/TOPS-20 system and the MULTICS system. Any modifications necessary to make SPICE a commercial product will be the responsibility of the partner.

Heterogeneity

Although we expect that the SPICE hardware and software efforts will be sufficiently exciting to capture much of the department's imagination, innovation, and design effort, SPICE will never be the only activity in the department. CMU-CSD is committed to a vigorous research and education program, which will not be curtailed in order to pursue SPICE. Wherever possible, we will encourage overlap of the pursuit of SPICE with research and education.

Likewise, SPICE hardware will dominate the department's facilities, but will never exclude other equipment. Other manufacturers' equipment will always have a place at CMU-CSD for several reasons:

1. Our present hardware has a useful life that overlaps the SPICE future. Some of our equipment has 5-8 years of remaining life.
2. Some computing needs of our research will require special-purpose computing. For example, we are working with CDC to design a high-speed computer for image-understanding applications. Such machines will exist only in small numbers, and often without general-purpose programming facilities.
3. Experimental hardware may be constructed, using commercially-available parts as a base. Our two multiprocessors, C.mmp and Cm*, have both used minicomputers (PDP-11's and LSI-11's

respectively) as the basic processor.

4. Gifts and grants from manufacturers or foundations may add considerable equipment to our environment. The Xerox equipment grant is a case in point: 17 Alto computers, a file server, a printing server, and Ethernet communication link have been given to the department. We expect considerable use of this equipment, and anticipate that our association with Xerox will continue.

Such heterogeneity must be tolerated by our SPICE design and, indeed, by any commercial personal computing product. The need to connect personal computers to other equipment in smooth ways is likely to be a persistent feature of this style of computing. The necessary heterogeneity in the department should provide insight into designing personal computing environments that can accommodate connections to other computing devices of all sorts.

Advantages to the partner

We summarize here the advantages that we see for the partner in entering into the SPICE venture with us.

The most important advantage is the development of a SPICE-like environment. In our view, this is analogous to the development of a sophisticated time-sharing environment around the PDP-10s by a small set of universities in the late sixties and early seventies. We believe the style of computing exhibited by SPICE will be the dominant "high technology" style of the mid-eighties. Not until the lessons of the SPICE style are learned, will the computer field be prepared to evolve the next style beyond that. Experience gained in this project will determine whether or not CMU-CSD and a partner will play significant parts in future developments. *We wish to emphasize that this proposal has to do with our relative positions in this development, not with whether such a development will occur.*

There is much to be learned about how to move from today's time-shared environments to the world of personal scientific computers, and about how best to exploit the opportunities of this new world when we get there. It may be several years before decreasing hardware costs make an environment like SPICE commercially attractive on a large scale, but it is important to begin learning about this new world now, before the economic event occurs. What CMU-CSD actually does for the partner is to blaze the trail toward this new environment: to produce the initial iteration of systems, to find out some of what works and does not work, to take some of the risks, and to move rapidly down this path.

An extremely important feature of SPICE is that it integrates the coming personal computing into a framework that is consonant with organized effort. The immediate response to the ability to have personal computers is the wholesale development of standalone micros. While there are important markets (e.g. recreational computing) where this sort of "line of least resistance" is probably appropriate, learning how to exploit the personal computer to attain more substantial computing goals is critical and difficult.

A further important advantage to the partner is that he obtains the efforts of creative first-rate computer scientists in the development of a highly marketable system, designed from the start to meet the needs of the scientific community. The partner harnesses not only our talent at implementation, but our ideas and judgement as well. Proof that such collaboration between industry and a university can be mutually beneficial may be illustrated by the advantage that has accrued to DEC from the university contributions to the PDP-10

development in the last decade. A joint SPICE venture will mobilize a community of over 100 PhD-level computer scientists and engineers (we include here senior graduate students) who will be deeply concerned with the partner's personal computing system, its problems and its future shape. This is a body of talent and expertise that would be hard for a partner to duplicate.

Finally, there are a number of benfits of a marketing kind that will come from the SPICE project, which will be open to public view. While we believe these benfits will have great value to our partner, we trust that the partner has more skill than we do in evaluating such benefits.

The venture requires that the partner bear almost all the cost of the hardware. Given the current expense of computers it might seem that CMU-CSD should bear some fraction of the cost. In fact, the total contribution we propose to make is probably more than the total cost of the hardware. More important, in terms of the success of this enterprise, the funds available to CMU-CSD to extend its computing facilities are limited. Funds that do not have to be paid for hardware will go instead towards the SPICE software effort. A significant fraction of the SPICE manpower will be supported by CMU-provided funds.

Advantages to CMU-CSD

This proposal constitutes a major commitment on the part of CMU-CSD, which carries with it high costs. We enter into it because we feel that these costs are offset by great advantages. The advantages are not entirely summed up in the total computing power of the SPICE hardware we obtain. We therefore indicate the reasons why we put forth this proposal at this time, in order that a partner may understand how we view the proposal.

Though CMU-CSD has a substantial computing environment (one large KL10, two large KA10s, two experimental multiprocessors, C.mmp and Cm*, and several experimental PDP-11/40 systems) it is currently completely saturated. Not only is it saturated, its level of sophistication has failed to keep pace with other advanced computer research organizations, especially in industry. Moreover, our computing needs continue to grow. Thus, at this time we need to do something substantial about our computing environment, both quantitatively and qualitatively.

It is clear to us that the days of the PDP-10 are now numbered. Though the PDP-10 system will no doubt remain useful for some time for limited growth and continued operation, it is no longer the obvious system of choice for extension into the future. Unfortunately, there is no good alternative to the PDP-10. The problem is not that there are no appropriate computing machines. Many powerful new engines are being developed. The problem is that of obtaining an appropriate software environment for future research.

We view with immense concern and trepidation the alternatives that face us in obtaining this new environment. In particular, there appears no way to avoid major software investments on our part, whatever course we choose. Leaving the PDP-10 world simply has its costs. Staying with it at this point saves the software costs, but spends our funds now on a path that will have to be aborted soon, when the new direction taken by the world of advanced computer research becomes manifest.

The current proposal is the best alternative that we can find to resolve this issue. It faces up to the immense

software investment that will be necessary to create a comprehensive new environment for our own work. But it makes that effort worthwhile by promising to provide an environment that will be useful through the late eighties and by offering the possibility of major contributions to the shape of the computing environment of the eighties.

CMU-CSD is much concerned about the problems of diversity, and of system and facility maintenance, both hardware and software. We do not desire to be responsible for software or hardware maintenance on idiosyncratic systems. Some of this is forced on us by the nature of research and advanced development, but it is a major source of cost and instability. The adoption of a SPICE world limits the diversity of physical systems, especially at the basic facility level. That many of the advanced software developments required in SPICE would find their way into partner-maintained systems is an immensely attractive aspect of the present proposal. We would expect SPICE to evolve continually to become less idiosyncratic and more dependent on standard software and hardware. (We see this as simply limiting the maintenance load, since new advances will continually create additional diversity and idiosyncratic maintenance problems.)

The emphasis on obtaining a large number of SPICE machines soon is not only the rational course for obtaining SPICE, it is essential if we are to commit our full energies to this course single-mindedly. We cannot overemphasize this. Gradual introduction of SPICE machines will have the effect of producing an environment that is dispersed over many types of machines with a much less focused (and much smaller) software effort. In blunt terms, if there is no assurance that we are building ourselves a major environment for tomorrow, the CSD community will not be able to convince itself collectively that it should muster the immense effort that is being proposed. This is not just a question of decisions by the senior people in CSD. In a university research community such as CMU-CSD, perceptions and commitments are arrived at and maintained independently. Thus the partner's commitment to a substantial set of initial SPICE computers constitutes a pre-emptive move that sets CMU-CSD fully on this course for the next few years.

Recent Accomplishments of CMU-CSD

SPICE is an ambitious undertaking, but we believe that it is well within the capabilities of our department. The Computer Science Department of Carnegie-Mellon University is one of the oldest and most distinguished computer science departments in the world, consistently ranking in the top three along with Stanford and MIT. We list here a number of recent accomplishments of CMU-CSD, as a rough guide to the level of ability that we will be able to bring to this task. This list makes no attempt to be exhaustive; rather, we highlight those accomplishments in areas related in some way to SPICE.

Multi-processor Systems:

- Designed a writable control store for PDP-11/40 system (the 11/40E), later used in the C.mmp system.
- Developed the first multi-mini-processor (C.mmp) with a non-trivial number of processor elements (16).
- Developed the first flexible object-oriented operating system for a multiprocessor environment (Hydra) and demonstrated its use.
- Developed a flexible, hierarchically structured architecture for multi-processor systems (Cm*) and demonstrated a prototype system with 50 processors.
- Developed an object-oriented operating system for Cm* (StarOS).
- Formulated and demonstrated language constructs for parallel processing (Algol-68 on C.mmp and Cm*).

Software Technology:

- Formulated one of the first programming languages for software development (Bliss).
- Developed a highly optimizing compiler for a system-development language (Bliss/11).
- Developed a modular software architecture for the design of a family of operating systems.
- Formulated concepts for a language (Alphard) with facilities for abstraction and verification of programs.
- Formulated the concepts of a production quality compiler-compiler.

AI, Speech, and Image Understanding Systems:

- Developed a highly efficient implementation for production systems (1977).
- Developed the first 1000 word vocabulary connected speech understanding system (Harpy, 1976) with over 90% sentence accuracy.
- Developed the first connected speech understanding system that works with male and female speakers, in terminal room environments, and capable of live spontaneous demonstration (Harpy, 1976).
- Developed the first 1000 word vocabulary speech understanding system operational on a low-cost mini-computer (LOCUST, 1977).

- Developed the first image segmentation algorithm to work on a wide variety of natural scenes.
- Developed a new instruction set in firmware useful for image applications.
- Designed an integrated image data base containing both symbolic and signal information.

Responses

This paper is not a request for formal proposals, but rather an invitation to join in a partnership to build SPICE. Companies interested in being a partner in the venture are invited to explore SPICE informally with CMU-CSD. Arrangements to discuss SPICE can be made by contacting:

Howard Wactlar
Director of Research Facilities
Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa. 15213

412-578-2571

Although initial discussions will be informal, it is likely that formal proposals will have to be prepared eventually by any companies interested in joining with CMU-CSD.

References

Although no single published document describes the SPICE environment we aspire to, most of the ideas in SPICE are borrowed from existing systems. This section attempts to point briefly to other reading.

Personal computing (although not personal *scientific* computing)

1. Alan Kay, "Microelectronics and the Personal Computer," Scientific American, Sept. 1977.
2. Alan Kay and Adele Goldberg, "Personal Dynamic Media," *Computer*, March 1977.

Personal computers

1. C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, "Alto: A Personal Computer," to appear in in D. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures, Readings and Examples*, second edition.
2. "PERQ," Three Rivers Computer Corp., 160 N. Craig St., Pittsburgh, Pa. 15213.
3. Alan Bawden, et al, "Lisp Machine Project Report," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, AIM 444, August 1977.
4. G. L. Steele, Jr., and D. A. Moon, "CADR," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, April 11, 1978 (MIT LISP machine report).

User interface through display

1. W. Teitelman, "A Display-Oriented Programmer's Assistant," *Proc. 5th Int. Joint Conf. Artificial Intelligence*, pp. 905-915, 1977. Expanded version available as Xerox Palo Alto Research Center publication CSL 77-3.
2. D. C. Swinehart, "COPILOT: A Multiple Process Approach to Interactive Programming Systems," Stanford Artificial Intelligence Lab. AIM-230, July 1974.
3. K. A. Lantz and R. F. Rashid, "VTMS: A Virtual Terminal Management System for RIG," TR 44, Computer Science Dept., Univ. of Rochester, Rochester, N.Y., May 1979.
4. G. Robertson, A. Newell, and K. Ramakrishna, "ZOG: A Man-Machine Communication Philosophy," Carnegie-Mellon Univ., Technical Report, August 1977.

Operating systems

1. B. W. Lampson and R. F. Sproull, "An Open Operating System for a Single-User Machine," 1979 Symp. on Operating System Principles.
2. D. D. Redell, et al, "Pilot: An Operating System for a Personal Computer," 1979 Symp. on Operating System Principles.

Graphics

1. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, second edition, McGraw-Hill, 1979.

2. See Alto paper, above.

Communication protocols and programming

1. D. R. Boggs, J. F. Shoch, E. A. Taft and R. M. Metcalfe, "Pup: An Internetwork Architecture," to appear in *IEEE Trans. Comm.*, January 1980.

2. J. Postel, "Internet Datagram Protocol - Version 4," IEN 80, USC Information Sciences Institute, February 1979.

3. J. Postel, "Transmission Control Protocol - Version 4," IEN 81, USC Information Sciences Institute, February 1979.

4. R. F. Sproull and D. Cohen, "High-level Protocols," *Proc. IEEE*, 66(11):1371-1386, November 1978.

5. J. A. Feldman, "High Level Programming for Distributed Computing," *CACM*, 22(6):353, June 1979.

Communication hardware

1. R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *CACM*, 19(7):395-404, July 1976.