CMU-CS-82-132

MISE: Machine for In-System Evaluation of Custom VLSI Chips

R. Bisiani, M. J. Foster, H. T. Kung, K.Oflazer

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

To appear in the Proceedings of IEEE 1982 Real-Time Systems Symposium

The research was supported in part by the Office of Naval Research under Contracts N00014-76-C-0370, NR 044-422 and N00014-80-C-0236, NR 048-659, in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539. M. J. Foster was supported in part by the Fannie and Hertz Foundation.

Abstract

The ultimate goal of any custom chip must be its effective use in a complete system. Integrating a chip into a system is usually a nontrivial task, especially for small research teams that do not have extensive hardware and software support. As a result, chips designed by universities and research laboratories are seldom evaluated beyond functional checking in isolation. *In-system evaluation*, namely the evaluation of chips in a realistic application system, is a crucial step in the development of a chip—without it, the impact of the custom chip on a system cannot be demonstrated adequately. Unfortunately, in-system evaluation has often been regarded (mistakenly) as a "tedious but simple" interfacing problem. This paper identifies some of the key research problems that one encounters in specifying, designing, testing and demonstrating a custom chip *in relation to the application system* in which it will be used, and proposes a system called MISE (Machine For In-System Evaluation) to be a solution to the issues raised.

1. Introduction and Motivation

With the advent of Very Large Scale Integration (VLSI) and the availability of multiproject chip, fast turnaround fabrication facilities [22] in universities and research centers, many researchers have begun to design special-purpose VLSI chips. Custom chips for performing computationally expensive inner loops such as filtering, inner products, and butterfly operations for real-time applications are being built regularly (see, e.g., [2, 3, 10, 18, 19, 23, 24, 28, 30]). The goal of these custom devices is to relieve performance bottlenecks in some larger system. Special-purpose chips must therefore be developed not in isolation, but rather in the context of a larger *target system*. Although simulations and analytical models are worthwhile, there is no substitute for measurements on a prototype of the target system. By means of *in-system evaluation*, designers can avoid deluding themselves about the usefulness of their devices.

Although desirable, in-system evaluation is difficult and expensive. Construction of a prototype target system with measurement facilities may be a larger project than the design of the chip itself, and calls for a different set of skills than those required by chip design. The system usually contains several hardware and software components other than the device to be tested. For example, if a custom sorting chip is designed (see, e.g., [29]), the target system must contain memories or disks to hold the data to be sorted, a general-purpose computer with software to control the sorting process, and hardware and software to interface the chip to the rest of the system. In this case, the target system is likely to be much more complex than the sorting chip.

In-system evaluation is desirable not only for the evaluation of a custom chip after it is built, but also for the specification of the chip in earlier stages of the design process. For this task, *chip emulators* at various levels of detail can be important tools for designers. A breadboard which closely models the device's actual

structure permits on-board probing; furthermore,an emulator breadboard provides faster and more realistic simulation than a software simulator. For instance, in the case of the sorting chip discussed above, chip emulation might show that for the specific data distribution at hand, a chip with a slightly larger on-chip memory would dramatically improve performance. Breadboarding a speech recognition chip for a voice-input layout editor may enable designers to evaluate the specification of the chip (i.e., information on its "data sheet") in a real user environment. For a special-purpose microprogrammable chip, chip emulation might indicate that it is crucial to support the execution of certain instructions directly in hardware to enhance the performance of the target system. This type of information can most easily be obtained through emulation of the chip in an environment that closely resembles the system in which it will be used. Like the implementation of a prototype target system, building an emulation breadboard for each individual custom chip is both difficult and expensive.

In-system evaluation could be made relatively easy and cost-effective, however, by some special systems that need only be designed once. In the following section, we sketch our concept of such a system.

2. A Possible Solution: MISE

We present the tentative architecture of a system called Machine for In System Evaluation (MISE). The goal of MISE is to allow designers of special-purpose VLSI devices to "easily" construct prototype target systems incorporating their devices, and to evaluate the devices and their impacts on the target system before and after they are built. In general, in-system evaluation of *multi-chip* custom devices is much more difficult than that of *single-chip* counterparts, and requires extensive hardware support. In this and the following two sections, we consider only the problem of in-system evaluation of single-chip custom devices. In Section 5, we will sketch a possible MISE system for in-system evaluation of multi-chip custom devices.

A MISE system for single-chip devices would include the following components, depicted in Figure 1:

- *Target System Host* (TSH): The TSH is a system that can host a variety of target system simulations and their interfaces to custom devices. Consider, for example, the evaluation of a VLSI chip for the control of joints of a robot arm. In this case the TSH would simulate arm movements, and interface the simulator with the chip in order to monitor or demonstrate the impact of the chip on the system. In particular, the TSH contains the following components:
    - A *general-purpose computer*, with memories and peripherals (such as graphics terminals and disks) that serves as the central controller for the TSH. It provides the software to model a target application system and to interface the custom device to the target system.
    - An *interface subsystem* for connecting the custom device to the general-purpose computer, including:
        - a set of memory modules to support the operation of the custom device, and to buffer

data and/or instructions between the general-purpose computer and the custom device, and

- *I/O channels* for real-time data acquisition by the device,and
- a *device controller* that handles data transfers between the custom device, the memory modules, I/O channels and the general-purpose computer.

It is expected that the designer of a custom device will have a good idea of how the device and the application system interact in the final system. The interface subsystem will provide mechanisms to model this interaction as easily as possible. For example, if the custom device has been designed to be interfaced to some standard bus, the interface subsystem will enable the designer to model the behavior of that bus. Furthermore, by integrating the interface subsystem into software systems running on the general-purpose computer, the custom device can be driven directly by test or simulation programs written in high-level languages.

- A *programmable "socket"* providing high speed physical connections for the custom device to communicate with the interface subsystem. The facilities at the pin-level interface of the socket are similar to those available on a sophisticated logic analyzer and pattern generator [13, 16].

  Programming the socket involves the specification of each "pin" either as an input or as an output (or both) terminal and the description of the encoding/decoding of the electrical signals sent or received by the chip in terms of higher level primitives. This encoding/decoding process will actually be carried out by a hierarchy of functions provided by the socket, the interface subsystem and the TSH software.

- *Custom Device Emulator* (CDE): The CDE is a fast programmable engine that can be plugged into the socket of the TSH and be programmed to emulate a *restricted family* of custom devices within various target systems. Its main function will be to help in the specification and evaluation of a custom device before any effort is spent on actually designing the device. The CDE contains the following components, or a subset of them:

  - a user interface to configure and monitor it,
  - memory and reconfigurable functional units to emulate a restricted class of custom devices, and
  - a programmable pin-level interface.

The CDE differs from software emulations because it is expected to be faster. It is obvious, however, that with any fixed structure, the CDE cannot possibly emulate all high-performance custom devices at their speed, especially those with highly parallel architectures. The goal of the CDE is to provide emulation support whenever it is practically possible. It is not the goal of the CDE, however, to support evaluation of the electrical characteristics of a chip (which is the logic analyzers' job); the CDE will simply behave within the timing tolerances of the chip that it is supposed to emulate.

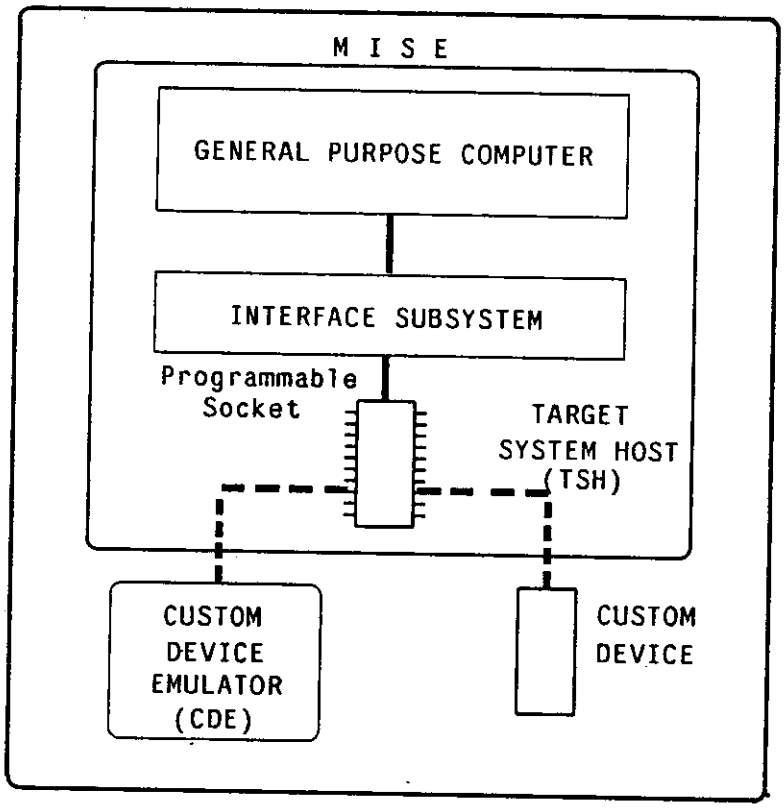

Figure 1: Functional structure of MISE.

We see that the TSH and CDE serve quite different functions; the TSH is a testbed for general target systems that incorporate special-purpose VLSI chips, while the CDE is a breadboarding system for some classes of VLSI chips. In particular, the function of the TSH does not depend upon that of the CDE and vice-versa.

Considerable experience has been obtained by LSI tester manufacturers in pin-level chip interfacing and fast test pattern generation [8, 13, 16]. Although the chip interface in MISE could employ techniques or components developed in this context, commercial testers cannot be used without change. They have features that are not necessary for MISE (e.g., DC testing) and lack other features such as programmability and integrability with a target system.

Some commercially available systems do address a very limited subset of the problem of in-system evaluation. Most notable among these are the so called *In-circuit Emulators* (ICE) provided by some companies to support the development of microprocessor based systems [15]. An ICE module is a fast emulator for a specific microprocessor. It provides an integrated environment in which hardware and

software development can be closely coupled. Extensive tracing and breakpointing capabilities are provided. During the development phase, the ICE is used in place of the microprocessor so that peripheral hardware configurations can be easily debugged. ICE tools provide only a subset of the proposed facilities of MISE. The CDE portion of MISE will provide aids for the specification and design of a variety of chips, as opposed to just one chip as in an ICE approach. In addition to being restricted to the emulation of only one single type of chip, the user of an ICE is still required to develop custom interfaces to the rest of the system. Furthermore, an ICE user is responsible for the software that integrates his custom hardware with the rest of the target system.

3. Two Application Examples

To illustrate the use of MISE, we describe the programmable recognizer array (PRA) [11] as an example of a custom chip that could benefit from in-system evaluation. Currently under development at CMU is a single chip that can be programmed (by laser fusing or software controls) to recognize strings generated by a given regular expression. A typical target system for this chip is a machine like that described by Haskin [12] for searching very large databases. In such a system, the chip would be used as a term matcher, examining data coming from secondary storage, and passing only data that met some primitive criteria to the rest of the system for further processing.

A typical system in which the PRA might be used is the text database searcher shown in Figure 2. To search the database in response to a query, the PRA's are programmed to look for strings, and the query resolver is programmed to combine PRA results. Each PRA receives data directly from the disk heads, and filters out data that is clearly irrelevant to the query at hand. Data that remain are passed to the query resolver, which combines PRA results. Finally, the twice-filtered data are passed to the host computer. For example, to search for sentences about databases, the PRA's might be programmed to recognize sentences containing the strings "data" and "base". The query resolver would filter out sentences in which the strings were combined in the wrong way (e.g.,"base line data"). Sentences which passed both of these filters would be sent to the host computer.

To show that the PRA is effective in such a system, one would like to test it within a prototype database system. In this case, the TSH could provide the secondary storage, data routing, and user interface needed for a database system, whereas the CDE could emulate the chip before it is built. Measurements of the performance of the system with and without the PRA (or its emulation) in operation could determine the best ways to specify and use the custom chip.

There is no effective substitute for in-system evaluation of the PRA term matcher. The data-dependent

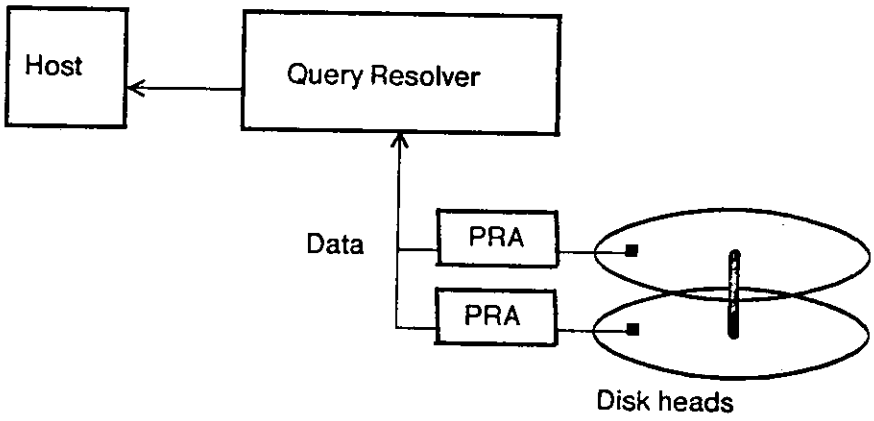

Figure 2: Database Search System

nature of database query evaluation makes analytical modeling impossible. Software simulations do not suffice either, since performance of the system as a whole critically depends upon the speed of the PRA. Simulated term matchers that cannot keep up with the disk I/O rates could bias any performance measurement, calling into question any conclusions based upon software simulation. Only in-system evaluation can provide a reliable estimate of the performance gains in this case.

As a further example of the use of MISE, let us examine the steps needed to evaluate a *speech recognition chip* of the kind described in [2]. Such a chip requires two kinds of real-time data streams:

- the "unknown utterance" represented as a set of feature vectors that are computed by sampling the microphone analog signal and then by applying a signal processing algorithm to extract "relevant" features of the signal and represent them as a sequence of vectors in time.

- "word templates" that describe each legal vocabulary word. Templates are sets of feature vectors that are computed by averaging and clustering the feature vectors obtained from a few pronunciations of the same word.

The generation of word templates requires a substantial amount of software to support the data collection and merging. When templates are available, they must be stored in such a way that they can be transferred to the chip in the right order and at the right speed. At the same time, part of the application system has to manage the analog data acquisition and perform the signal processing on the unknown input. The feature vectors obtained by processing the unknown input must be fed to the chip at the right time and in synchronism with the templates. Finally, the recognition system has to be integrated with some realistic user interface and possibly support some plausible application so that the evaluation is representative of a real case.

The components that we have just described require a design and implementation effort that is likely to be greater than the effort to design and fabricate the chip. The software is relatively complex and difficult to

debug because the correctness of its results is hard to evaluate independently of the rest of the system. Moreover, the system requires some specialized hardware in order to acquire the analog data and to feed the recognition chip with the necessary information in real time.

Two kinds of non-MISE ad-hoc solutions would be possible in this case. The first solution is to generate the software on a general-purpose system and then build some specialized hardware to interface the chip with the general-purpose system. Despite the existence of standard interface cards for general purpose systems, interfacing may still require extensive custom engineering. At the very least, software drivers will have to be written and debugged. (In a recent experience at ESL, a subsidiary of TRW, at least four man-months of effort were spent in linking a systolic processor to a VAX-11/780 using a DR11-B interface module [6, 31]). A second solution is to build a microprocessor based system that would act as a controller for the chip and that would take care of input and output data streams. The complexity of the hardware that would have to be built would be lower in this case but the effort to develop the support software would be substantially higher because of the poor programming environment that a small, ad-hoc microprocessor based system would provide.

On the contrary, MISE would support the programming of all the software in a comfortable environment. At the same time, it would eliminate the need for specialized hardware, because the interface subsystem and the socket would be able to handle the chip I/O. To use MISE, the designer is required only to describe the behavior of the chip. This description should overlap substantially with the specification of the chip that the designer has to provide in any case.

4. A Scenario for In-System Evaluation

Using MISE, designers could obtain rapid feedback on the impact of their chips on the systems they are intended for. The design and evaluation process of a custom chip will typically include the following steps, as depicted in Figure 3:

1. Conceptualize and specify the custom chip.

2. Develop the prototype target system software including the programs for the CDE and TSH needed for in-system evaluation of the chip. (Note that this stage is purely a software effort for the designers. They specify and design the prototype target system.)

3. Based on insights gained from the preceding step, iterate 1 and 2 and then finalize the specification of the chip.

4. Design, build and test the chip using other tools whose user interface and data representation is compatible with MISE.

5. Connect the working chip to MISE.

6. Run the prototype system on MISE with software developed earlier at step 2 and measure (and demonstrate) performance.

Among all the steps, step 5 usually receives the most attention in the research community. It is probably fair to say that most of chip development work in the research community has been on the design and implementation of individual chips. Research on systems such as MISE should provide insights for the other steps, which deal with in-system evaluation.

5. In-System Evaluation of Multi-Chip Custom Devices

Although single-chip custom devices are used in many applications, there are cases where multiple chips must be used to obtain the necessary performance. This includes systems such as systolic arrays (see e.g., [18]) that use many copies of the same chip, chips with different functions, or both. Because of the excessive computational requirements, the emulation/simulation of multi-chip devices in real time cannot be achieved by a general-purpose tool of reasonable cost. Nevertheless, simulation using a MISE-like system may still be effective even though it may be one or two orders of magnitude slower than real time.

The simplest solution to in-system evaluation of a multi-chip device would be to scale the target system down so that it would be sufficient to consider a reduced version of the custom device having only a small number of custom chips. In this case, one of the chips could be emulated in the CDE (at a later time when the chip is built, a copy of the actual chip could be used by plugging it into the programmable socket) and the remaining chips in the custom device could be simulated, at the *highest* possible level, by the general-purpose computer in the TSH. This approach is limited to applications that can be meaningfully scaled down to a point where the simulation of part of the custom circuitry by software in the TSH has a reasonable performance. A more powerful, but more difficult and more expensive solution, sketched in Figure 4, shows an expanded MISE system incorporating a new special hardware unit, called the *System Simulation Machine* (SSM). The purpose of the SSM is to speed up the simulation that was carried out by the general-purpose computer in the above solution. The improvement in speed, which can be of one to two orders of magnitude, is crucial to make MISE a useful system. In general, we expect the SSM to be a high speed programmable processor, providing:

- fast components such as ALU, RAM, multiplier, etc.,
- fast context switching (so that the SSM can effectively simulate more than one chip by multiplexing), and
- large memories with flexible addressing mechanisms.

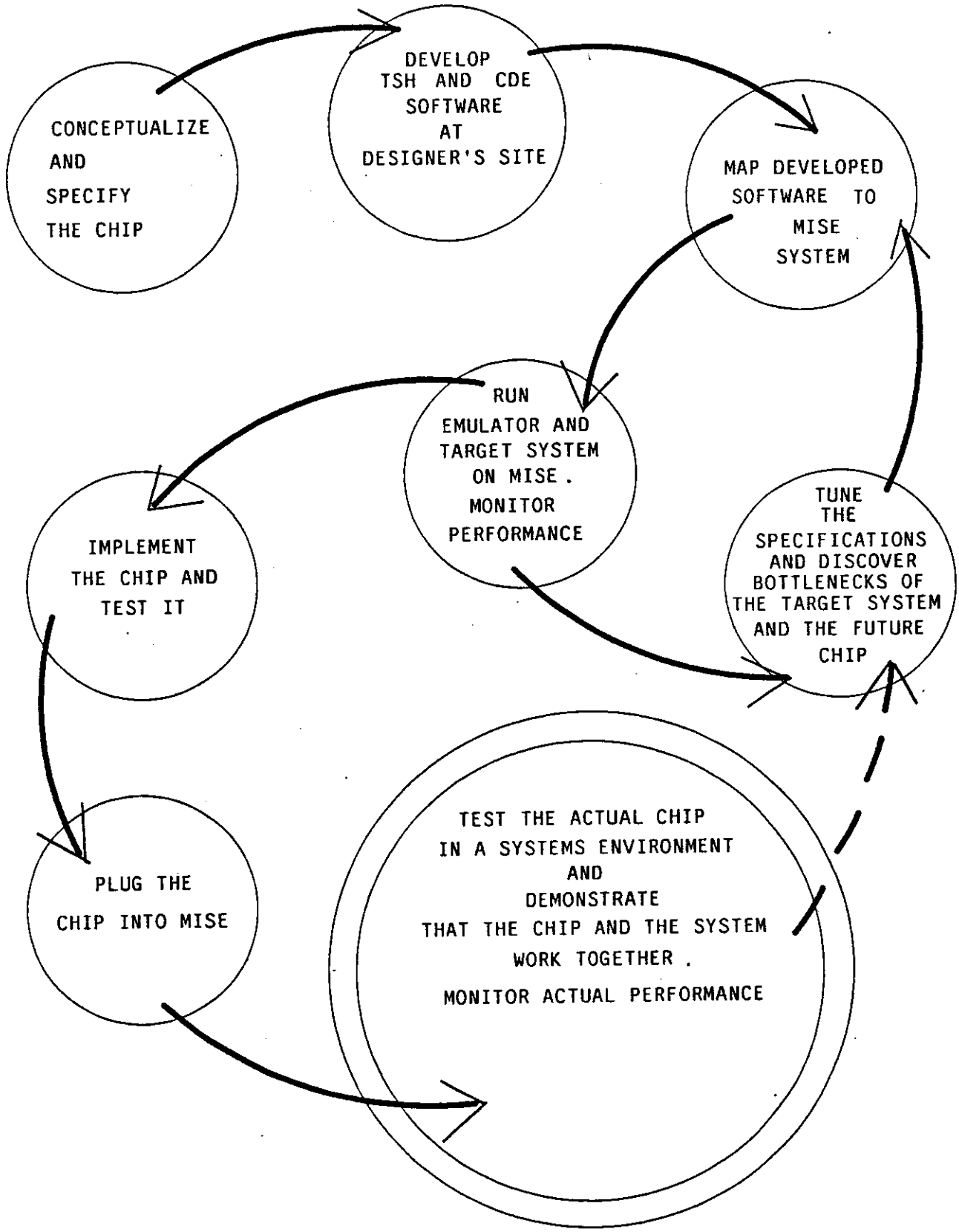

Figure 3: System Design Process Involving MISE.

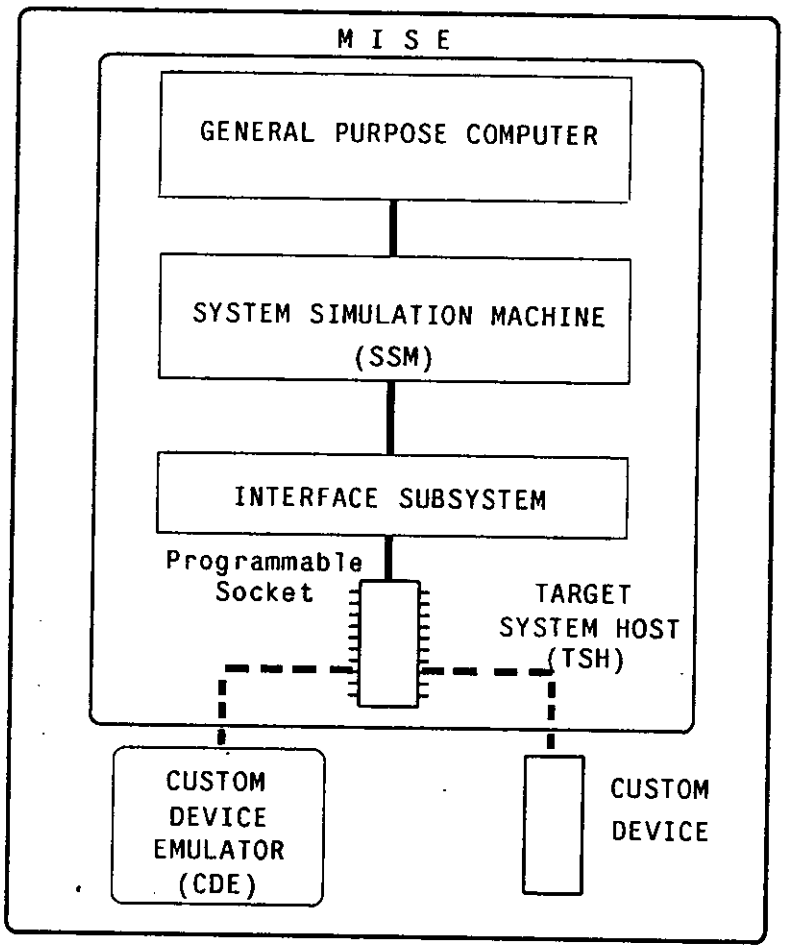

Figure 4: Functional structure of a MISE system
for in-system evaluation of multi-chip custom devices.

The scenario for in-system evaluation described in Section 3 would be modified as follows for multi-chip devices. The designer would first simulate the target system using the TSH, with the SSM simulating the function of the custom device. Iterating this step, the designer could evaluate various methods of partitioning the custom device's task among the custom chips and various interconnection schemes for connecting the chips. At this stage, each custom chip could be described at a high level by a module in the program executed by the SSM. Later, the designer could tackle the internal specification of each of the custom chips separately, program their description on the CDE, and run the whole system by substituting the high-level description of the chip used by the SSM with a different module that interfaces with the CDE. In most cases the use of the CDE would not improve the overall speed of the simulation/emulation but would allow the designer to experiment with the internal organization of the chip and measure the performance in an environment that is as close as possible to a real target system.

To illustrate the use of such an extended MISE system for in-system evaluation of multi-chip custom devices, we consider the *programmable systolic chip* (PSC) that is being developed at CMU. The ISP architecture of the chip has been finalized [9], and the chip has been laid out in September 1982. This chip will serve as the building block for a variety of systolic arrays (see, e.g., [18]). By using arrays consisting of tens or hundreds of copies of PSC, we expect to get orders of magnitude improvements in speed for applications including:

- Convolution-like operations in signal and image processing,
- Encoding and decoding for error-correcting codes,
- Encryption and finger-print calculations [17, 27],
- Number theoretic transforms,
- Disk sorting.

Consider the application to Reed-Solomon error-correcting codes [21] as an example. Suppose that each codeword consists of 224 information symbols followed by 32 check symbols and each symbol is an 8-bit integer; using the Reed-Solomon scheme, errors involving no more than 16 symbols can be corrected. We estimate that by using a linear array of 112 PSC chips the Reed-Solomon decoding can be performed with a throughput of 10 million bits per second. Encoding is much easier; it requires only about 30 PSC chips to achieve the same throughput. As far as we know, the fastest existing Reed-Solomon decoder with the same characteristics uses about 500 chips but achieves a throughput of only 1 million bits per second. The performance of the PSC-based system is largely due to architectural innovations; for example, a highly effective systolic array has been developed for implementing the extended Euclidean computation, the most difficult task in the decoder implementation [7]. It is therefore desirable to implement a Reed-Solomon decoder using the PSC chip in order to demonstrate that architectural innovations can make a big difference in this application. Using the MISE system, the following approaches can be taken:

- If there is a sufficient number of working copies of the chip available, then a complete array of PSC chips, as required by the application, may be constructed as a board with a well-defined interface to the external system. The board can then be connected to the MISE "socket" as a custom device and evaluated as a single-chip system. In this case, the effort spent in constructing and debugging the multi-chip device would be high, but the results of the in-system evaluation would be very realistic since the whole custom system would be running in an environment close to its intended environment.

- When there are not enough working PSC chips available, a different approach would be to use the SSM to aid in the evaluation process. In this case a single copy of the PSC would be connected to the socket (or during the chip development phase, the CDE simulating the chip may be connected), while other copies would be emulated at the functional level by the SSM hardware. In order to perform this emulation, a very fast but relatively simple processor in the SSM would

multiplex itself on a memory containing the states of each PSC chip to be emulated. The SSM would also interface the functional emulation to the actual chip in the socket so that the whole system (consisting of the SSM , the interface subsystem, and the chip in the socket) would function like the actual "multi-chip" custom device, but running much slower. For a 112 chip PSC array, such a configuration could conceivably run around 100 times slower than real time assuming the SSM can emulate a single chip's function in real time. This would still be two to three orders of magnitude faster than an optimized software simulation we wrote for the same system on the VAX-11/780.

6. Research Areas

The scenario we have presented in the previous sections requires the solution of a number of challenging research and engineering problems. Some of these problems are important and interesting in several other contexts, but they will all have to be tackled before a MISE-like system can be built.

1. *Custom device characteristics that affect the design of MISE.* Characterizing and classifying the custom chips that MISE could support is an important issue since it largely determines the design space for MISE. Although the concept of providing special-purpose devices has been around for some time, VLSI implementations of such devices have started only recently. Custom chips should be classified according to criteria such as

   - whether they are programmable or not,
   - whether they need other chips to assist in a systems environment ,
   - whether a multiplicity of them connected in a certain configuration are needed in a working system (e.g., a systolic array),
   - whether they are instruction-execution machines or data-driven machines,
   - whether they are control intensive or data intensive,
   - amount of parallelism,
   - the number and kind of input and output ports required, and
   - the speed of operation.

   We are not aware of any previous study of this kind although findings from this study can be of interest to the general community of custom VLSI designers.

2. *The specification and implementation of interfaces.* In order to provide custom device designers with a means of easily interfacing their chips with the TSH hardware and tailoring the CDE to emulate these devices, it is crucial to identify an easy, formal way to specify the interface behavior of the custom devices. Hardware description languages provide a valuable tool for this problem. One specific notation for describing I/O hardware and its behavior is the SLIDE language developed at CMU [25]. Also, ISP [4] and PMS systems have been applied successfully to the description of a UNIBUS based system [5]. We feel that compatible notations should be chosen to specify both the CDE, SSM and the various interfaces. The main concern in choosing a

notation is the ease with which the designer is able to describe an interface, since a notation that is too complex to use would render the whole MISE approach ineffective.

3. *The structure of the CDE.* The level of emulation on the CDE is an important issue. To achieve fast simulation, the CDE might emulate only the I/O behavior (i.e., as a black box) of the chip. If the internal structure of the chip is to be examined, then it may be desirable to emulate the actual circuitry of the chip on CDE. For the latter case, how do we obtain the required speeds and parallelism? To emulate highly concurrent and regular structures such as systolic arrays, the CDE may incorporate modular, extensible arrays of identical programmable processors much like the PSC chip. Another possibility is to "scale" a target system so that only a "scaled down" version of a custom device has to be emulated. The question then is: "What makes a target system scalable?"

   Although there has been some work on computer architectures tailored to simulation [1, 20, 26], we do not know of any existing architecture that would be directly applicable to the CDE.

4. *The structure of the system simulation machine* (SSM). Many of the design considerations that apply to the CDE also apply to the SSM. The most obvious difference between the SSM and the CDE is that the SSM must be tailored to simulate the functionality of many chips simultaneously at the highest possible level, while the CDE must emulate one single chip and its pin-level input/output behavior. The decision on the class of target systems and chips to be supported can provide some idea of the possible structure and functionality of the SSM. The effectiveness of the SSM depends on how much of the physical characteristics of the system (e.g., the behavior of the interconnection between chips) can be safely ignored.

5. *The integration of design tools.* Eventually, design aids for chip specification, design, testing and in-system evaluation should all be integrated in a single system. A consequence of this integration is that much of the software can be shared by all phases so that a MISE-like system could grow into a machine for in-system *development*, rather than just for in-system *evaluation.* Little experience is currently available, however, on the integration problem [14].

6. *The usefulness of MISE.* A designer will choose to use MISE only if MISE is easy to use and powerful enough for his target system and custom device. The general questions of how to make MISE easy to use and to characterize applications for which MISE is suited are key issues.

At Carnegie-Mellon University, we have begun to study these issues. The plan is to build an experimental prototype MISE system within two years.

7. Summary

We have attempted to describe the motivations and requirements for in-system evaluation of custom VLSI chips. The need for tools that help in-system evaluation will be greater in the near future in view of the rapid increase of opportunities of using custom chips in various application systems. We anticipate a strong imbalance between the effort that goes into the chip itself and the effort that is needed to use the chip in a complete system. For example, we estimate that the ARPA MOSIS (multiproject, fast turnaround) facility now supports about 300 chip designs every year; in a few years this number is likely to be over 1000. This

means that many application systems that use these chips will have to be built and debugged at a great expense of time and manpower.

In-system evaluation of custom chips is not adequately assisted by design systems and tools available today. As pointed out in the preceding sections, several research and engineering issues have to be solved before a cost-effective in-system evaluation tool can be built. These issues are not easy but we must face the challenge of building an in-system evaluation environment now. Otherwise we will pay high prices in the future for repeatedly integrating custom chips into systems using ad-hoc methods.

Although many issues are still open, we have sketched a possible solution, namely a tool (MISE) tailored to aid the design, construction and evaluation of prototype target systems employing custom VLSI chips. Even if MISE-like systems can effectively support in-system evaluation for only a quarter of, say, the ARPA MOSIS designs, the contributions of these systems will be enormous.

8. Acknowledgements

We thank C. Ebeling, A. Fisher, and E. Frank of Carnegie-Mellon University for their comments on this paper.

References

[1] Abramovici, M, Levendei, Y.H. and Menon, P.R.
A Logic Simulation Machine.
In *Proceedings of the 19th Design Automation Conference.* IEEE, June, 1982.

[2] Ackland, B.,Weste, N. and Burr, D.J.
An Integrated Multiprocessing Array for Time Warp Pattern Matching.
In *The 8th Annual Symposium on Computer Architecture*, pages 193-203. ACM SIGARCH, April, 1981.

[3] American Microsystems Inc.
Fast Fourier Transformer.
In *MOS Products Catalog*, pages 2.90 - 2.91. American Microsystems Inc., 1980.

[4] Barbacci, M.R.
Instruction Set Processor Specifications (ISPS): The Notation and Its Application.
IEEE Transactions on Computers C-30(1):24-40, January, 1981.

[5] Barbacci, M.R., Djordjevic, J.M. and Hosler, B.W.
A PMS Level Notation for the Description and Simulation of Digital Systems.
In *Proceedings of Computer Hardware Description Languages.* 1981.

[6] Blackmer, J., P. Kuekes and Frank, G.
A 200 MOPS Systolic Processor.
In *Proceedings of SPIE Symposium, Vol. 298, Real-Time Signal Processing IV.* The Society of Photo-optical Instrumentation Engineers, August, 1981.

[7] Brent, R.P. and Kung, H.T.
Systolic VLSI Arrays for Polynomial GCD Computation.
Technical Report, Carnegie-Mellon University, Computer Science Department, May, 1982.

[8] Chalkley, M.J.
Trends in VLSI Testing.
In *1979 Test Conference.* IEEE, October, 1979.

[9] Dohi, Y., Fisher, A., Kung, H. T. and Monier, L.
PSC Architecture.
VLSI Doc., Carnegie-Mellon University, Computer Science Department, May, 1982.

[10] Foster, M.J. and Kung, H.T.
The Design of Special-Purpose VLSI Chips.
Computer 13(1):26-40, January, 1980.
Reprint of the paper appears in *Digital MOS Integrated Circuits*, edited by Elmasry, M.I., IEEE Press Selected Reprint Series, 1981, pp. 204-217. A preliminary version of the paper, entitled "Design of Special-Purpose VLSI Chips: Example and Opinions," also appears in *Proceedings of the 7th International Symposium on Computer Architecture*, pp. 300-307, La Baule, France, May 1980. .

[11] Foster, M.J. and Kung, H.T.
Recognize Regular Languages With Programmable Building-Blocks.
In Gray, J.P. (editor), *VLSI 81*, pages 75-84. Academic Press, August, 1981.
The final version is to appear in *Journal of Digital Systems.*

[12] Haskin, Roger Lee.
Hardware for Searching Very Large Text Databases.
PhD thesis, University of Illinois at Urbana-Champaign, 1980.

[13] Hentz, M.
An Advanced Pattern Generator for Memory Testing.
In *1980 Test Conference.* IEEE, November, 1980.

[14] Hon, R.
The Hierarchical Analysis of VLSI Designs.
VLSI Memo V073, Carnegie-Mellon University, Computer Science Department, December, 1980.

[15] Intel Corporation.
The 8086 Family User's Manual.

[16] Kapps, C.A.
Four Approaches to LSI Testing.
In *1979 Test Conference.* IEEE, October, 1979.

[17] Karp, R.M. and Rabin, M.O.
Efficient Randomized Pattern-Matching Algorithms.
Technical Report TR-31-81, Center for Research in Computing Technology, Harvard University, 1981.

[18] Kung, H.T.
Why Systolic Architectures?
Computer Magazine 15(1):37-46, January, 1982.

[19] Kung, H.T. and Song, S.W.
A Systolic 2-D Convolution Chip.
In Preston, K., Jr. and Uhr, L. (editor), *Multicomputers and Image Processing: Algorithms and Programs*, pages 373-384. 1982.
An extended abstract appears in *Proceedings of 1981 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, November 11-13, 1981, pp. 159-160.

[20] Lineback, J.R.
Logic Simulation Speeded with New Special Hardware.
Electronics 55(12), June, 1982.

[21] McEliece, R.J.
Encyclopedia of Mathematics and Its Applications. Volume 2: *The Theory of Information and Coding.*
Addison-Wesley, Reading, Massachusetts, 1977.

[22] Mead, C.A. and Conway, L.A.
Introduction to VLSI Systems.
Addison-Wesley, Reading, Massachusetts, 1980.

[23] Nishitani, T., Kawakami, Y., Maruta, R. and Sawai, A.
LSI Signal Processing Development for Communications Equipment.
In *Proceedings of ICASSP80*, pages 386-389. IEEE Acoustics, Speech and Signal Processing Society, April, 1980.

[24] Oflazer, K.
Design and Implementation of a Single-chip Median Filter.
Technical Report, Computer Science Department, Carnegie-Mellon University, May, 1982.

[25] Parker, A.C. and Wallace, J.J.
An I/O Hardware Descriptive Language.
IEEE Transactions on Computers C - 30(6), June, 1981.

[26] Pfister, G.F.
Introduction to the Yorktown Simulation Engine.
In *Proceedings of the 19th Design Automation Conference.* IEEE, June, 1982.

[27] Rabin, M.O.
Fingerprinting by Random Polynomials.
Technical Report, Center for Research in Computing Technology, Harvard University, 1981.

[28] Raibert, M.H. and Tanner, J.E.
A VLSI Tactile Array Sensor.
In *Proc. International Symposium on Industrial Robots.* Paris, France, 1982.

[29] Song, S.W.
On a High-Performance VLSI Solution to Database Problems.
PhD thesis, Carnegie-Mellon University, Computer Science Department, July, 1981.
Also available as a CMU Computer Science Department technical report, August 1981.

[30] Treleaven, P.C.
VLSI Processor Architectures.
Computer Magazine 15(6):33 - 45, June, 1982.

[31] Yen, D.W.L. and Kulkarni, A.V.
The ESL Systolic Processor for Signal and Image Processing.
In *Proceedings of the 1981 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 265-272. November, 1981.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CMU-CS-82-132	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): MISE: Machine for In-System Evaluation of Custom VLSI Chips	5. TYPE OF REPORT & PERIOD COVERED: Interim
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): R. Bisiani, M.J. Foster, H.T. Kung, K. Oflazer	8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0370, NR 048-659, NR044-422, N00014-80-C-0236, F33615-81-K-1539
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA. 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Arlington, VA 22217	12. REPORT DATE: August 1982
	13. NUMBER OF PAGES: 19
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)