A Preliminary Report on the Caltech ARPA Tester Project

by

Erik P. DeBenedictis

Technical Report #4061

April 12, 1980

Computer Science Department

California Institute of Technology

Pasadena, California 91125

sponsored by

Defense Advanced Research Projects Agency

Copyright, California Institute of Technology, 1980

Acknowledgements

This report describes many ideas that have originated in discussions between me and my advisor Chuck Seitz. We have tried to polish these ideas as much as possible to make them suitable for written presentation. Hopefully we have been successful with some of the ideas, but we certainly have not been with all.

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under contract number N00014-79-C-0597.

1.0 INTRODUCTION

For the non-industrial research engineer the importance of testing is often recognized too late. The day when an integrated system could be tested with logic designed the afternoon after the chips are returned from the production line is past. Testing today is done by a computer. In the future, however a mere 'computer' will be inadequate to perform testing efficiently. It will be necessary to construct a test instrument to efficiently execute the specialized testing strategies developed for various kinds of integrated systems.

In a similar fashion, it is no longer possible to scratch out a test pattern for testing a chip on a piece of notebook paper. Instead, the services of a computer are required to calculate the complex test patterns and hold the mass of test information. Computers will be unable to perform this task without considerable thought by the designers toward making the system testable.

We seek here to forewarn by discussing the issues and trends in testing, and to forearm by providing a test instrument.

1.1 Function Of A Test Instrument

Many strategies have been developed for testing chips. All of them have a common characteristic. They all involve applying

voltage levels to the chip's inputs and observing its outputs. This implies the existence of a machine to interface the test information to the electrical levels of the chip. We call such a machine a tester.

The complexity of the tester may vary greatly. At one extreme the testing of a microprocessor designed without a specific testibility mechanism may require a fast machine with a great deal of storage. At the other extreme, for a chip of similar complexity if a self-testing design methodology were used, a test machine no larger than a hand calculator might suffice. The design of the tester is therefore affected by the popular design methodologies.

In this note we will consider first the design of a test instrument aimed at testing methodologies that appear to be now gaining popularity. We do this without considering either their actual merit or their details. These considerations are hoped to yield a tester design that will be able to accomodate to likely design methodologies of the future. Later on, we will discuss the actual aims and fundamental problems associated with testing. We will attempt to discover the flavor of likely testing methodologies from fundamental principles.

2.0 TEST INSTRUMENT DESIGN

Certain obvious considerations about the nature of testing can be made in order to develop a general idea of how a tester should be designed.

It is expected that the complexity of the testing process is becoming very great for state-of-the-art chips, and that this trend will continue. Testing machines will therefore be called upon to process increasingly greater amounts of information; and they will be judged on how efficiently they do this. There is one fundamental bottleneck in the testing process that we must exploit to the fullest. This bottleneck is the information bandwidth at the pins of the chip. One conclusion that can be drawn from this is that a successful tester must be capable of applying test patterns to a chip at the greatest rate that the pins can tolerate. We believe this is the most important theme today.

2.1 Internal Tester Electronics

It may be difficult, but it is not impossible, to build a tester capable of driving test chips at their maximum speed. The internal speed of the integrated circuits that need to be tested is increasing rapidly now, whereas the speed of off-chip communications is not. Since testers will for some time be non-integrated systems there is a potential problem with the tester being fundamentally too slow. This imbalance can be

combatted by either using the fastest electronic logic for the tester, or utilizing such design techniques as parallelism and pipelining. Today, both suggestions have equal merit. The fastest ECL available is sufficiently faster than any high-density logic that a tester so built could easily be fast enough. Similarly, slower logic can be used in a more complex design to have the same effect. The speed of today's fastest ECL is near the maximum for non-integrated parts, however. Therefore, in the future, it is evident that parallelism and pipelining will be required also. To avoid a tester design that will become obsolete quickly we should include all the necessary parallelism and pipelining now.

2.2 Modular Tester Structure

The tester structure that we are presenting has a number of parts, none of which will be entirely specified. Different users may have different requirements with respect to these unspecified features. For example, some users may wish to test relatively high voltage logic, requiring unusually complex front-end electronics. Some users may not need high voltage capability and do not wish to complicate the machine with such a feature. We suggest that the tester be made modular where possible. This means, for example, that the front-end should be made detachable from the other parts of the tester so that a user can put on a front-end tailored to his specific needs.

2.3 Tester Front End

The part of the tester closest to this bottleneck is the linear electronics that interfaces to the chip's pins. It is necessary that these electronics be sufficiently fast and sufficiently flexible so that the application of test patterns is not impeded. It is not necessarily important that these electronics be infinitely fast or capable of resolving large numbers of voltage levels. As chips become larger the number of internal gates increases faster than the number near the periphery. In other words, one expects that a progressively smaller fraction of a chip's behavior can be characterized by detailed analog measurements at its pins. It is inevitable therefore, that design techniques will be adopted that do not require such measurements. As this happens such analog measurements at the pins of a chip will become less important.

2.4 Test Pattern Generators

The requirement of high performance implies that there will be special hardware in the system to generate efficiently commonly used test patterns. Ideally the tester would have parts individually tailored to the common test strategies. In today's terms, there would be a part that is good at generating test patterns for memories, one for processors employing LSSD, and a part for testing systems with internal state available through a bus structure.

At present we do not know what testing strategies will become popular in the future. This does indeed make it difficult to engineer sensible equipment. A solution to this dilemma is to build a tester that has a facility for adding these special parts as they become sufficiently well understood. Such a solution would mean that there would be no end to the design process for the machine, but also that it would become obsolete more slowly.

The straightforward way to design a pattern generator is to have a large memory that stores the test pattern. The width of this memory is the number of pins on the chip and its depth is the number of steps in the test. This memory is stepped through to apply its contents to the pins. In this form, the idea is ludicrous. Usable test pattern generators can be viewed as variations on this simple structure, however. Such variations involve using less memory and altering its contents during the course of the test, or having hardware to generate portions of patterns directly from algorithms, bypassing any memory.

2.4.1 Standard Pattern Generator -

We believe that there is one type of pattern generator that will be adequate for testing most chips. We suggest that such a pattern generator be a standard part of the tester. This generator will be designed to apply test patterns repetitively, with minor changes each time. This pattern generator will

employ a pattern memory of relatively small size. This memory will be somewhat dual-port. It will be accessed to have its contents applied to the chip under test, and it will be accessed to have minor changes made in its contents. As will be shown later on, this has the features necessary to test memories and to test processors with state available via internal busses.

2.5 General Pattern Generation And Application Structure

We recommend that the tester include the following structure. The linear electronics and fixturing apparatus be connected to a large bus. This bus will have sockets for pattern generation units that generate specific types of test patterns. The bus structure will wire-ORed such that each unit may drive or monitor any group of pins.

As another feature of modularity, the number of channels supported by the tester need not be fundamentally fixed. Some parts must have this fixed, such as the front-end and the bus, where a physically contiguous part has logic for each channel. Other portions of the machine can be constructed on a per-channel basis, using as many parts as are necessary. The tester could be designed so that standard parts, and a special front-end and bus could be used to make a tester with an arbitrary number of channels. An illustration of these ideas is Fig 1.

The pattern generators require information dependent on the details of the chip to operate. This information will come from another portion of the tester. This portion will have storage, and may do some processing of its own.

2.6 Feedback From The Test Instrument

The fundamental function of a tester is to return to the user information about whether a chip works. In production testing this information is merely a go/no go indication. If anything is wrong with the chip it is disgarded. In other cases it may be desired to obtain more information about why the chip malfunctioned. In some cases "exploratory testing" may be desired. This is where the chip performs quite differently from the way it was intended and the designer wishes to manually characterize it.

We propose two mechanisms to achive this. One mechanism indicates that there is something amiss, and the other allow analysis of the problem. The tester will both apply signals to the pins of the chip and compare the signals from the chip with specified values. If all the comparisons succeed then the chip is functional. If any comaprison fails then there is something wrong and analysis can begin. Therefore, one mechanism is to have the testing procedure stop when an output from the chip is different from its expected value.

Once it is known that a particular test failed there are generally two ways to analyze why it occurred. One is to look backwards in time from where the failure became recognized and examine in greater detail the outputs of the chip. The other way is to perform additional tests on the chip to localize the problem. To accomplish this we propose that the tester keep a pin record. This will be a memory that records the state of every pin each test step. The memory will only save the most recent occurances, as memory size permits.

It will be the responsibility of the controlling computer to apply other tests and manage interaction with the designer.

2.7 Test Information Storage

As chips become larger the number of bits required for a test becomes greater. It is necessary that the test instrument have storage elements of sufficient performance to drive the pattern generation hardware. This memory will have different performance requirements than those normally encountered by persons familiar with general purpose computers. In particular, the test information will be accessed in a sequential fashion, rather than in a random access fashion. The familiar doctrine of having various storage media with varying speed/capacity ratios is probably not applicable. Instead, we believe that the proper storage mechanism will need mostly a high volume, high bandwidth, sequential medium, that will be accessed at a

constant rate.

Some simple arithmetic should show what type of storage medium will have to be used. The fundamental consideration is that the storage medium used should have the time to read its entire contents roughly the same as the time of a test. This argument is independent of the volume of information. If, for example, the time to test were on the order of a half hour then the proper medium would be a magnetic disk. However many disks would be required to provide the necessary bandwidth would be used. It would be unwise to use a faster memory, such as semiconductor, because the cost would be higher and the performance would be the same. It would be also unwise to use a higher capacity, slower, memory because it could not possibly be all utilized in the course of the test.

The actual time of a test depends upon the use of the chip. For production testing it is likely that test times will be limited to tens of seconds for economic reasons. For testing very low volume chips, an acceptable test time could be much longer. It is true, however, that design methodologies have been developed that guarantee a tens of seconds test time for the largest imaginable chips.

Given a test time on the order of several tens of seconds a magnetic disk would be inappropriate because only a small fraction of its contents can be accessed in that time. In present technology, MOS RAM is the proper choice. The entire

contents of a large MOS RAM*system (10 Mb) can be easily read in the necessary amount of time. In the future magnetic bubble or CCD storage systems may become competitive with present MOS RAM systems for this application.

We suggest that the entire storage for the tester be organized as a central high-bandwidth memory, as opposed to distributing it among the pattern generators. This memory will have ports similar to DMA channels to the various test pattern generators. This has several advantages over the alternative of having the storage distributed where it is used. One advantage is in the simplicity of debugging tests. A general purpose computer connected to this memory can do the loading of tests from disk, interpret the ultimate output of the tester, etc. This will reduce the cost of the custom pattern generators, as they would require only an interface to a storage medium, not storage themselves. In addition, the necessary volume of storage can be reduced because the storage in the main tester would be dynamically allocated, as required.

2.8 Interfacing Issues

In proposing a test instrument for use by the ARPA community it is important to consider user interface issues. Such a test instrument will likely cause a great deal of effort to be devoted to automatic test generation. In addition, the instrument may be used by persons remote from the actual device.

It is important that the interface design issues be known and well understood by those persons who will use them.

2.8.1 Remote Operation -

It is an implicit goal of this tester design that persons remote from any fabrication/testing facility be able to use the instrument. This means that that person must be able to construct his test information reliably, and in a transportable fashion, in the absence of the tester. In addition, he must be able to manage his design in such a way that chips can be installed in the tester without requiring the physical presence of the designer. Certain characteristics of the test instrument are constrained by these goals.

2.8.2 ARPA-net Interface -

There must be an interface to a common data communication medium. This medium will be without question the ARPA-net. The tester design must specify a form for this ARPA-net data. The design must be robust against design changes that obsolete this form. In a later section we will propose a standard test language of this form. A complete overview of the tester, as presently developed, is illustrated in Fig 2.

2.8.3 Standardization Of The Chip Interface -

In addition, there is a physical interface to the tester that must be standardized. Such issues as how to distribute power and what type of electrical loading to apply to pins must be addressed.

We suggest that the tester be constructed with some specified number of channels, all the same. In general, the permutation of the tester's channels to the logical groups of pins on a chip will be accomplished in software. Normal channels will have an interface that can both drive and monitor the state of a pin. Impedances will be very low for driving and very high for monitoring. In order to distribute power and adjust impedance levels the chip will be mounted in a semi-custom designed fixture. This fixture will likely be a PC board. The board will connect the tester channels and power supplies to the pins of the chip. Discrete components can be put onto the traces to adjust impedance levels. If some users were to agree on a standard configuration for power supplies and impedance levels, they could all use the same fixture. An illustration of the chip interface is Fig 3.

2.9 Signal Levels

Different types of design may utilize different voltage thresholds, or may need different numbers of signal states. There are actually two issues here, although they are

intermingled. Some of the problem can be easily resolved by using digital-to-analog converters freely throughout the tester. These might generate the power supply voltages and threshold voltages, and would be software controllable. These would be independent of the main structure of the tester, and this is an issue whose resolution can be delayed. The other issue, of how many input and output levels to provide, will have an effect on the bit width in nearly every part of the tester. The concept of the test instrument can have the number of signal levels as a parameter, but once a tester is physically designed the number of signals levels is permanently fixed.

As was previously pointed out, an increasingly small portion of a chip is testable by analog measurements at its pins. Trends today are toward interconnections that are as voltage and delay insensitive as possible. We suspect that the importance of having many more degrees of freedom in the tester than there are in the internal logic of the chip is very small. Today the vast majority of chips have internal logic with two or three states, i.e. ones, zeroes, and sometimes tri-state. There are presently some logics involving three active signal states that appear to have promise. We suggest that an appropriate number of signal states is three. This number is larger than will be necessary in most cases in the forseeable future, but not so much larger that it could be said to be extravagant.

3.0 SPECIFICATION OF TEST PATTERNS

This chapter is devoted to the design of the tester from the software point of view. The tester's structure will be described as it might be seen by a remote user.

3.1 Graphical Representation Of A Test

To gain an insight into the nature of the tester's task, consider a graphical representation of a test. There are two dimensions to this graphical representation. On one dimension are the different pins of the chip, and on the other is the sequence of test steps. The content is symbols representing the state of a pin during a particular step of the test. Time is loosely related to the sequence dimension in that time progresses in one direction, but not at a uniform rate, along that dimension. The symbols control what signals are applied to the chip and how the signals coming from the chip will be interpreted. Examples of signals are "apply a 1 to this pin", "expect a 0 on this pin and flag an error if one doesn't occur", "delay advancement to the next step in the sequence until this pin is a 1", etc.

3.1.1 Informational Volume Of A Test -

This graphical representation contains a great deal of information. A quick estimate can be obtained. If a test of a 100 pin part is 30 seconds long, and the average rate of step sequencing is 10 MHz, and it takes 6 bits of information to describe a symbol, then we are referring to a picture with 180 billion bits. It would be infeasible to build a test instrument that has this much internal storage, therefore a major function of the test instrument is to compress this information.

3.1.2 Redundant Information In A Test -

In typical testing this graphical representation would have much redundant information. Our understanding of testing state-of-the-art chips is that there are steps that are applied to the pins to cause the internal state of the device to become available, and similarly set. Typical testing involves repeatedly setting the internal state, causing the device to be cycled for a short while, and for the internal state to be read and compared with the known correct value. We believe that a typical test may be 99% internal access sequence.

From these considerations we suggest that the tester should have a memory whose width supports a symbol for each pin. This memory can be operated by two sub-parts of the tester. On one side is the part that reads out to the chip under test, and on the other is a mechanism that writes test patterns into the memory.

3.1.3 Images Of Internal Registers -

The internal access sequences can be viewed graphically. They will typically involve applying a great many worthless signals, and a few containing information. If, for example, an internal register were loaded in parallel at some point then there would be a set of points in a row, or a horizontal image, of the register in the graphical access sequence. Similarly, if there were a register being loaded serially then there would be a vertical image of that register in the sequence. This phenomenon is illustrated [fig 1]. To make use of this observed regularity we propose that the tester be able to apply a stored pattern repeatedly, with changes.

3.2 Repeated Application Of Test Steps

This leads us to define commands to the tester that cause particular test steps to be applied to the chip. These commands will cause certain entries in a memory to be applied to the chip. The memory, whose depth need only be several hundred deep, is altered between these commands. This idea is illustrated in Fig 4.

3.3 Logical Pin Permutations

It is necessary that the test instrument be able to make the small changes in the stored patterns in an efficient manner.

In particular, it should be possible to instruct the tester to make a parallel entry into the test matrix with an arbitrary bit permutation. Along these lines we suggest the following: the tester should have the ability to define logical pin groupings. Each grouping will map a tester bus of limited size, perhaps 16 bits, to an arbitrary group of pins on the chip. There will be perhaps 64 such groupings. As an example of use, it may be desired to apply a count sequence to a group of pins on the chip. At some point it is necessary to perform the arbitrary permutation from a counter to the pins of the chip. A more mundane use of the permutor is to merely act as a demultiplexer to allow the wide test pattern memory to be loaded from the more reasonably sized main store. The general flavor of the implementation of the permutor is illustrated in Fig 5.

As presented, this feature is limited to some number of permutations. The number of simultaneously defined permutations is determined by the hardware, but since the permutations will be software definable it will be possible to define a permutation immediately before use. This extends the feature to an arbitrary number of permutations with a sacrifice in efficiency.

These considerations lead us to propose commands that cause pin permutations to be loaded, and that some other commands will have a field that specifies which pin permutation to use.

3.3.1 Serial Access And LSSD -

The concept of a serial image of a register is very similar to that of the parallel image, however it is believed to be used in a very different way. At present the only known methodology that uses serial images is IBM's LSSD. In this case, however, there is only a single pin through which the test data flows. This indicates that perhaps serial images require less generality. This, coupled to the fact that it is considerably more difficult to write such images into a parallel memory, leads us to suggest that efficient application of serial images be left to a custom pattern generator.

3.4 Direct Pattern Alteration Commands

The considerations just presented lead us to propose tester commands that directly alter the test pattern memory. In general these commands would specify multi-bit data and a pin permutation. The data bits get permuted and loaded into the pattern memory.

3.5 Additional Redundancy

Certain types of testing have test patterns that include count or shift sequences. Memory testing normally requires that successive addresses be accessed, and testing of iterative arrays can be accomplished efficiently by shifted test patterns.

In tests such as these the tester structure already developed will reduce the number of bits required for a test by orders of magnitude. The resulting information, in these cases, will contain redundant count or shift sequences. This is illustrated in Fig 5. This redundant information can be eliminated by an additional level of processing. A machine that can generate count and shift commands from higher level commands is in order. This machine can be viewed as a filter. The machine is in the stream of commands from the storage to the pattern generators, it interprets some commands that are directed toward it and expands them into others.

3.5.1 Custom Second Level Hardware -

There is no reason to believe that counting and shifting are the only important functions that can be performed at the filter level. It is possible that unforseen sequences such as ternary counting or incrementing CRC characters may become popular. To accomodate unforseen schemes we can allow custom filter hardware in the same way as custom pattern generation hardware. We suggest that there be a box with the shift and count functions. In addition, there would be empty slots where a user could insert a custom board. This also implies that the commands to be interpreted by this box be designed to accomodate additional functions. A summary of proposed command types is Fig 6.

3.6 Exchangability Of Hardware And Software

The preceeding discussions have presented a machine with the facility for adding additional hardware to make certain types of testing more efficient. It is desirable that the lack of special hardware not be fatal to the performance of any test. In particular, it should be possible to perform any test with only the standard hardware, although at a slower rate. This requires that the standard hardware be sufficiently general to apply an arbitrary test pattern. The hardware presented has this property. In addition, there must be software that can expand test specifications for special purpose hardware units into specifications for the available hardware. Ideally, the test information would be transmitted to the tester in a very high level form, and the tester's software would fit this to its hardware.

4.0 TESTING PHILOSOPHY

The manufacturing process for integrated circuits is to transfer the information present on the masks to a chip of virgin silicon. If a manufactured integrated circuit has the mask information properly incorporated into its structure then the circuit will work. If there were a major flaw in the chip, such as that caused by dust, then there would be parts of the chip where the pixels on the mask were very different from what was on the chip. At the other extreme, if there were a very

small non-fatal imperfection in the chip, much smaller than pixel size, then that error would not constitute incorrect information transfer because pixels imply a statistical averaging over some area.

It will certainly be argued that, in any chip, there are always some areas where correct transfer of the mask pixels doesn't matter. A blank area between not-so-well fitting parts can have anything in it without affecting the performance of the chip. Similarly, some layers are less sensitive to errors than others. The implant layer in NMOS need only be defined over a small fraction of chip area. These arguments indicate that correct operation is easier to obtain than getting every pixel correct. On the contrary, the goal of designers is to minimize the blank area between not-so-well fitting parts, or in general to maximize the sensistivity to errors in these ways. It is also true that the implant mask is only one of several, and some of the rest are very critical to operation of the chip. Therefore, the previously developed notion that a chip will work only if it is pixel-for-pixel the same as the mask is only slightly over-restrictive.

4.1 The Testing Task

This viewpoint has implications for testing. It is now sufficient to verify, by some means, that all the pixels on the manufactured chip are correct. This is an informationally much

simpler process than some schemes that have been proposed. The number of systems on a chip that can hold N transistors is O(k**N). The number of systems that are a general interconnection of N transistors is O(N!). On the other hand, this is a more detailed test method than those currently in use. Such methods may assume a maximum of a single fault over the entire chip, and that the fault may only be a gate stuck at one or zero.

The really important question is how do we verify that the patterns on a chip are pixel-for-pixel correct? The general procedure is to calculate what functional effect an incorrect pixel would have on the chip's functioning, and to then verify that a pixel is correct by performing that function and observing the result. Unfortunately, there are quite a few pixels to check in a large system and, in a thoughtlessly designed system, the procedure to check each one may be so long that the entire test time becomes astronomical.

Thoughtlessly designed chips tend to require long sequences of signals applied to the chip in order to determine arbitrary information inside. Design techniques have been developed that guarantee that there is quick and easy to find procedure for testing for any flaw. Such techniques presently work only for limited classes of systems. They may work sufficiently well for those systems that there never need be further thought given to those systems, however [LSSD]. We believe that the classes of systems for which this does work are sufficiently limited that

further thought into these testing strategies is warranted.

The notion developed previously about internal access sequences and images of internal state variables can serve as a guide for testing strategies. With many design styles the length of the internal access sequence can be determined approximately by a cursory inspection of the internal architecture, and the necessary number of elementary tests is often bounded by allowable fan-in, fan-out, and propogation delay.

4.2 Alternate Testing Strategies

All existing testing strategies work by causing the pixel information to be electrically transfered from the chip to where it can be examined by a tester. Present trends are to maximize efficiency by increasing the amount of pixel information available at the pins of a chip relative to the amount of set-up information that must be supplied. This places a fundamental limit on the informational complexity of the test process. All of the pixel information that went into making the masks must emanate from the pins of the chip. If it were possible to test without actually getting all of the test information from the chip then this maximum might be only a local maximum and vastly more efficient testing strategies might exist.

4.2.1 Internal Test Mechanisms -

Strategies that test without transfering much information from the chip do exist. Such strategies are not very profound, so one will be described in an example. It is well known that testing of certain classes of systems by means of application of random patterns will locate most errors in few tests. Possible figures are 99% of all errors in 200 tests. Now, consider building into a chip a pseudo-random number generator that will generate several hundred tests. Also there will be built onto the chip a mechanism for taking the result of the random tests and condensing it into a single number, or signature, by means of some algorithm. This algorithm will be characterized by a large Hamming distance. The chip can then test itself. The only task left for the tester is to compare the number generated as a result of the test with the number that good chips have.

Self-test mechanisms such as the one described require that there be some additional logic present on the chip. If the amount of this logic is on the order of the rest of the system then the method is economically infeasible. In addition, it is often desired to have some information as to why a chip did not work, and this if not directly available if the pixel information never leaves the chip.

4.2.2 Self Testing Logic -

There are other profoundly faster ways to do testing. Consider a chip composed of gates. Given the ability to transform the gates into AND-gates and OR-gates the chip can self test itself for signal conductors stuck to a power supply. The test procedure is to change all the gates to OR-gates and apply all ones the the inputs. This should cause all internal signal conductors to go the the one state in the order of a microsecond. If the gates are then transformed to AND-gates then the presence of any conductor stuck-to-zero will cause many internal gates, and at least one output, to switch to a zero state. Stuck-at-one faults can be similarly detected. Were there a way to construct gates in such a manner that they could be transformed from their normal function to AND- and OR-gates and that failure of any internal component would have the effect of a stuck-at fault, then we would have a potential testing scheme. A six transistor CMOS design has been devised with this property.

5.0 APPENDIX - DESCRIPTION OF INFORMATION FORMS

Information on a multi-conductor bus will be indicated as follows:

{ A B C ... }

This indicates that there is a multi-conductor bus that at some time has the information A B C ... simultaneously on its conductors. There will certainly be timing conductors, which are not described.

Multiple cycle information may be represented as follows:

{ A ; B ; C ; ... }

This indicates that there is a multi-conductor bus that can accomodate at least the information A B C ... one at a time. The information A B C ... will be transmitted over the bus in consecutive cycles. Timing information, although not described explicitly, may be present to identify what part of a cycle the bus is in.

5.1 Information On The Tester Bus

Form: { Sp ... S1 S0 }.

where the Sn are pin state symbols. p is the number of pins supported and is a parameter of the tester.

5.2 Pin State Symbol

Form: { T1 T2 V1 V2 }.

where the T's determine the pin type on that step and the V's determine the logic value assiciated with that pin.

T1 T2	type of pin	V1 V2	logic level

0	0	inactive pin	0	0	inactive
0	1	tester drives chip	0	1	zero
1	0	compare chip output and indicate failure if not in proper state when step advances	1	1	1/2 middle state
1	1	wait for chip output to be in proper state before advancing step			

5.3 Primary Change Information

Form: { M P A S15 S14 ... S0 }.

where M is a mode code indicating the type of change, P is an index indicating the permutation to be used, A is the address in the primary memory that will be changed, and S15 S14 ... S0 are sixteen symbols that will be written into the memory.

mode	function
Change memory.	Store the sixteen symbols S15 ... S0 into primary memory location A with permutation P.
Change permutation.	Change the permutation with index P according to the information in A and S15 ... S0.

5.4 Primary Sequencing Information

Form: either { M X }

or { M B E }.

where the first form is used when sequencing is not desired, and the X indicates another command (primary change information.) The second form is used to specify sequencing and is identified by the mode code.

Sequencing information	
Primary memory steps beginning with B and ending with E are applied to the chip.	

5.5 Shift / Count Filter Commands

Form: either { M X }

or { M D }

or { M A P S15 S14 ... S0 }

where the first form is used when invocation of the shift / count function is not desired. The other forms are identified by a mode code indicating shift or count. The A P S15 S14 ... S0 are primary change information.

Shift / Count Commands	
Load	Load the shift / count register with information from the D field.
Change and Count	Change address A of the primary

	memory using permutation P. However, do not use the data portions of S15 S14 ... S0, instead substitute the integer in the accumulator. Then increment the accumulator.
Change and Shift	Same as above but shift the accumulator.

5.6 Macro Expander Filter Commands

Form: either { M X }
or { M N ; C1; C2; ... Cn }
or { M N K }

where the first form is indicated by a mode code not involving the macro expander. The second form defines a macro with name N, or executes a macro.

Macro Expander Commands

Define	Store the commands C1 C2 ... Cn into a memory with label N.
Execute	Execute the macro named by N K times.

Portion of Tester

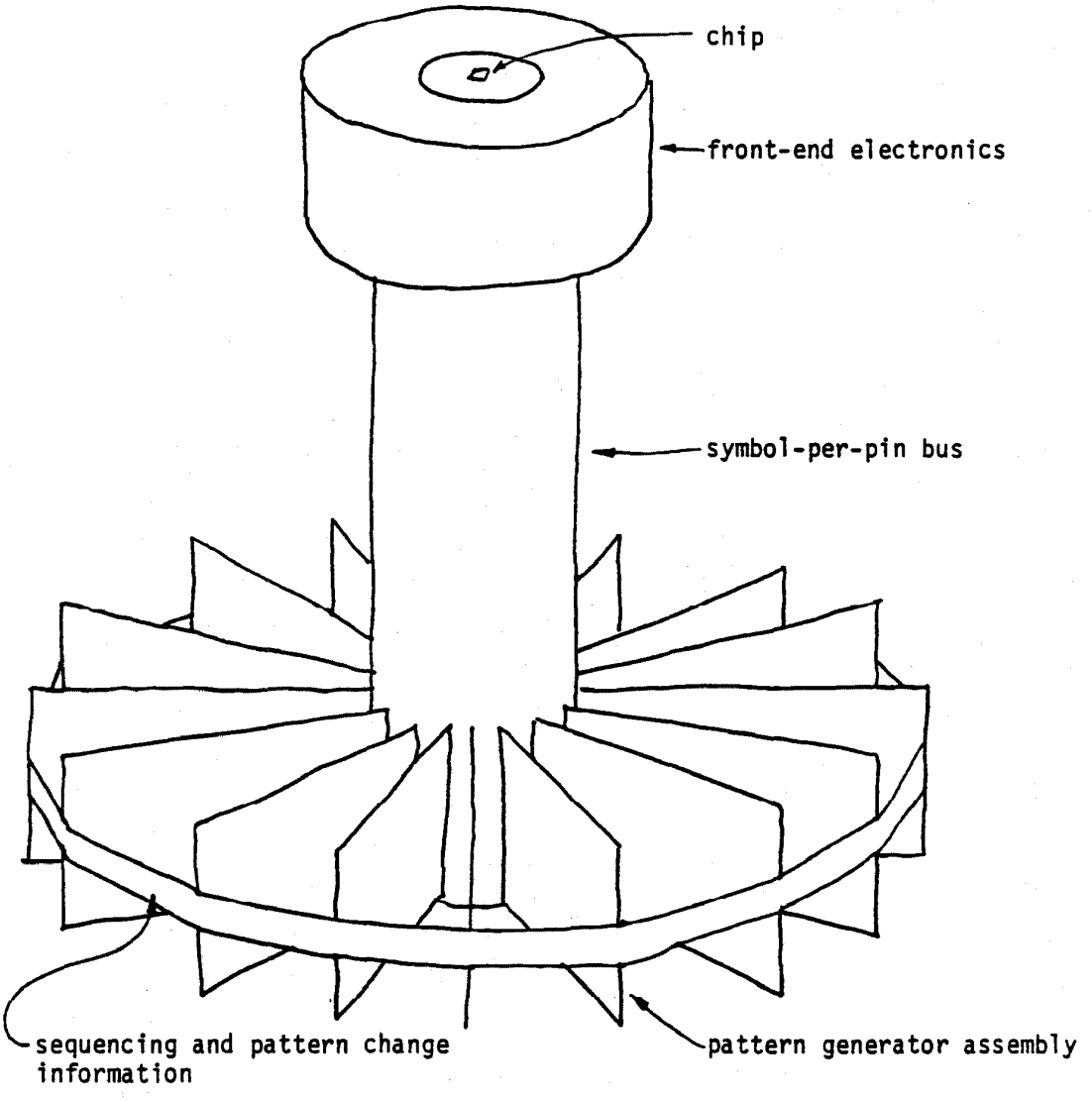

Figure 1

Overview of Tester

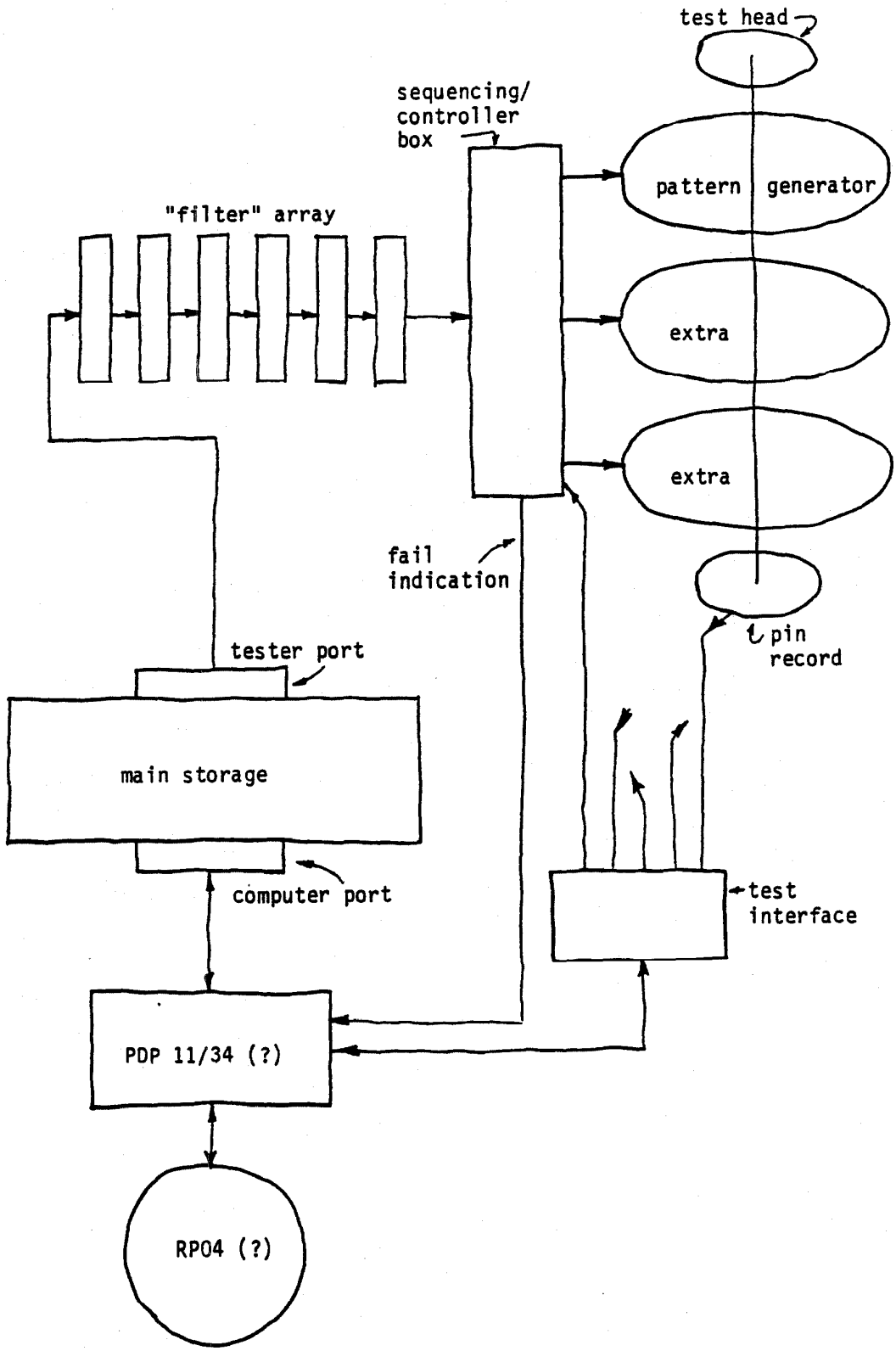

Figure 2

Fixturing

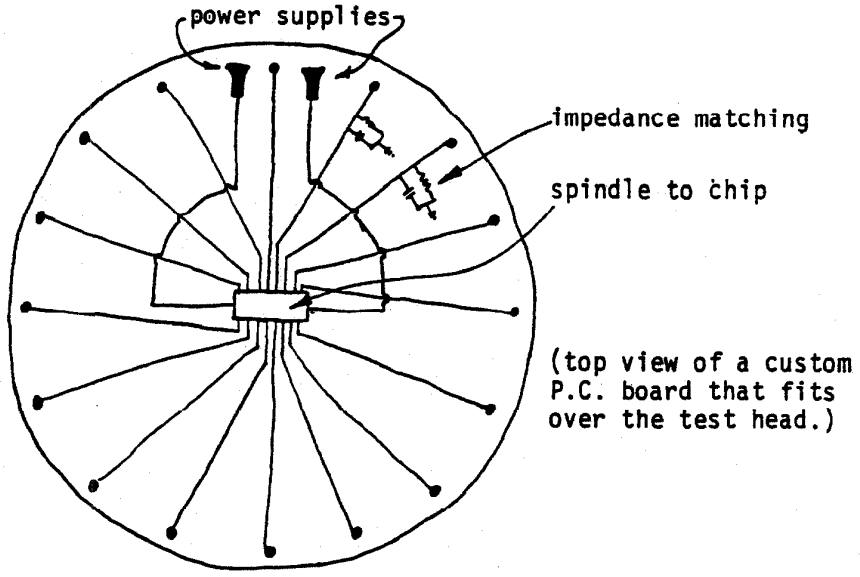

Figure 3

Graphical View of a Test

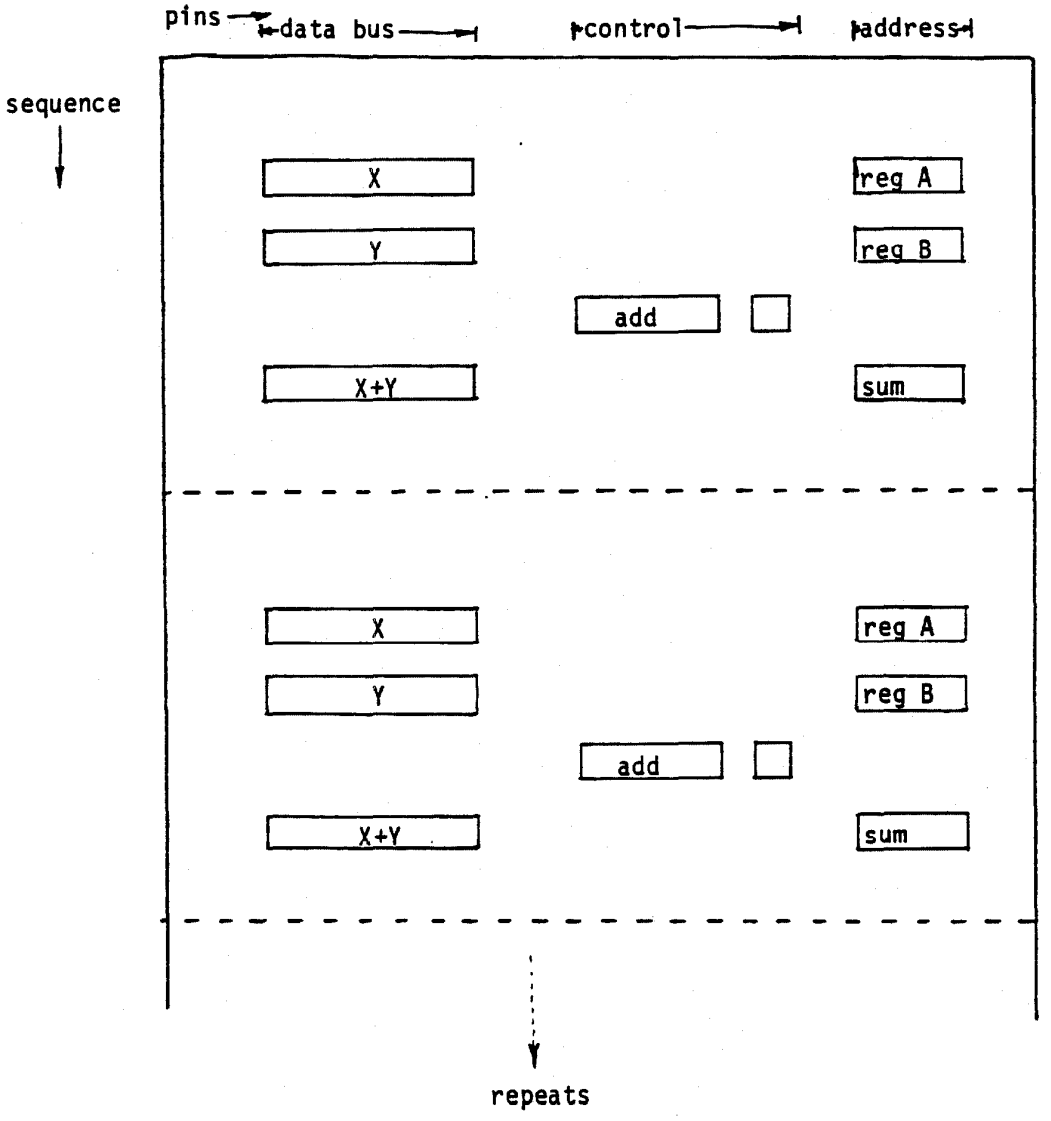

Figure 4

Pin Permutations

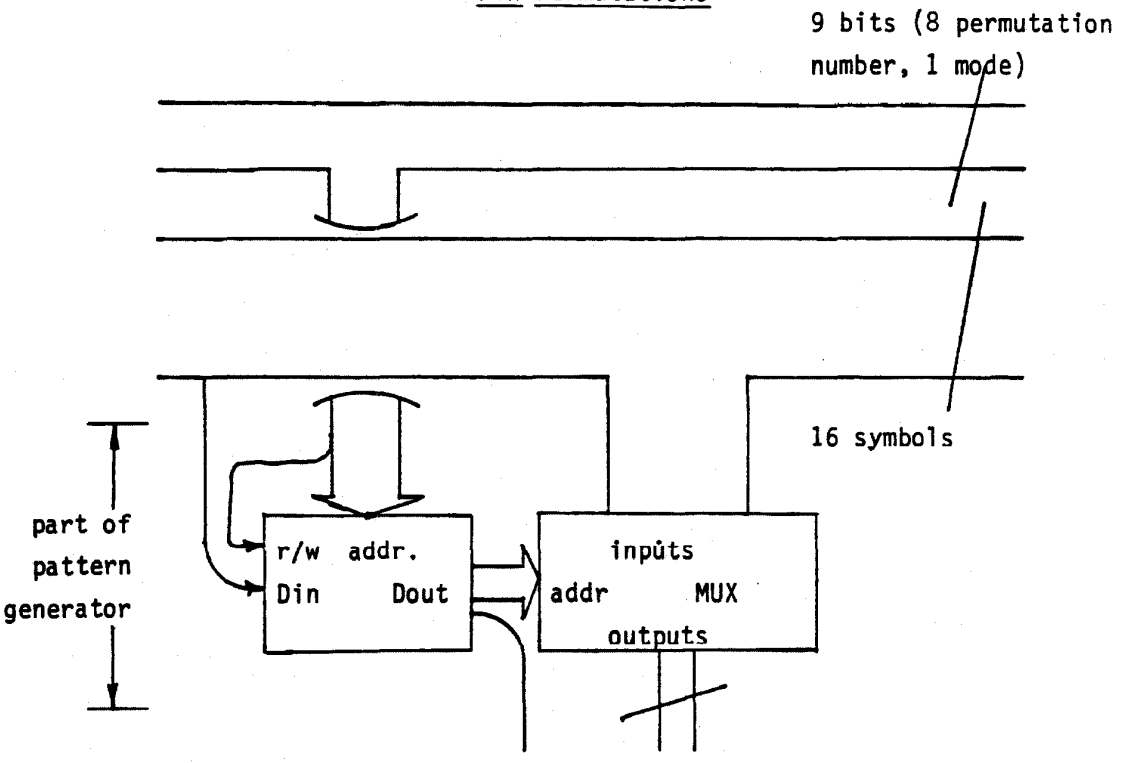

Figure 5