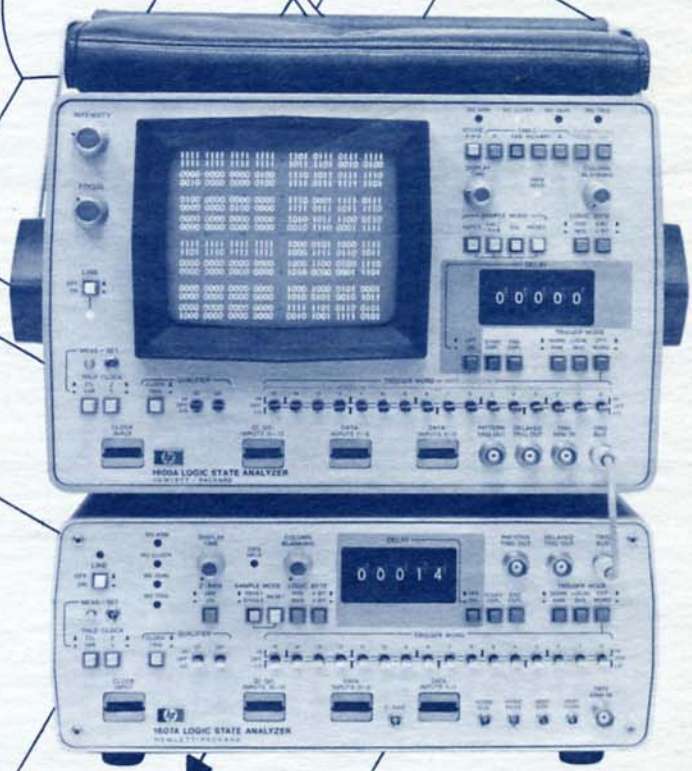
APPLICATION NOTE 167-13
DATA DOMAIN MEASUREMENT SERIES

The role of logic state analyzers in microprocessor based designs.



For More Information, Call Your Local HP Sales Office or, in U.S., East (201) 265-5000. Midwest (312) 677-0400. South (404) 436-6181. West (213) 877-1282. Or, Write: Hewlett-Packard, 1501 Page Mill Road, Palo Alto, California 94304. In Europe, Post Office Box 85, CH-1217 Meyrin 2, Geneva, Switzerland. In Japan, YHP, 1-59-1, Yoyogi, Shibuya-Ku, Tokyo, 151.

THE ROLE OF LOGIC STATE ANALYZERS IN MICROPROCESSOR BASED DESIGNS

by Bruce G. Farly
Hewlett-Packard Company
Colorado Springs, Colorado

INTRODUCTION

Not too many people question the fact that we are in the midst of a revolution, a so-called microprocessor revolution, and many pages have been devoted to trying to explain the whys and the future trends of this revolution. We at the Hewlett-Packard Colorado Springs Division, have been more concerned with the new measurement problems that are proliferating along with this revolution. Certainly the problems of logic state flow analysis were not originated by the microprocessor, but public awareness of the need for new measurements and the pressure for the instrument manufacturers to provide them is mounting.

The case for data domain measurement instrumentation, and in particular, logic state analysis, has been more than adequately documented in the literature. In particular, Charles House has written a fine treatise on logic state analysis (Application Note 167-4) which clearly defines not only the need for it, but also the differences and synergisms between logic state analysis, logic timing analysis and the more traditional time domain or oscilloscope analysis. This paper will assume that the reader is familiar with the basics of logic state analysis and deal specifically with some more complex problems, especially real time problems, as they relate to microprocessors.

INSTRUMENTATION IN THE MICROPROCESSOR DESIGN CYCLE

It is proper to question at what point in the design cycle instrumentation enters the picture. As a generalization it enters as soon as hardware is present be it a partially working breadboard or some combination of simulation/emulation equipment and the designers own hardware. A considerable variety of aids are available to the microprocessor designer, but when the troubleshooting problems become more real time in nature, more subtle in their interactive hardware/software ideosyncrasies, a real-time, non-interactive measurement tool is invaluable.

Illustration is still the best way to make a point. So, let's consider some fairly common problems to solve, often very troublesome without suitable measurement capability.

TIME AND OTHER PARAMETERS VS. STATE FLOW

In this first example, consider a case where the designer wished to measure the overall cycle time of an addressing system in a microprocessor. His case involved address buffering, ROM chip selection logic, and 3-state buffering of the data bus. Doing a theoretical calculation of the access time of this addressing scheme yielded at best only hypothetical results. What he desired was to know just how fast his system really worked. To this end, he instrumented the experiment as shown in figure 1. In place of the microprocessor a stimulus was applied that was fully controllable in both frequency and state value so that addresses could be generated at a variable rep rate. In order to measure the response of the system, a data monitor was needed that read the memory output data and indicated when that data was no longer correct. For this he used a logic state analyzer as a response monitor. (This example illustrates stimulus/response testing in a digital world analogous to the stimulus/response testing so commonly used in the analog world.)

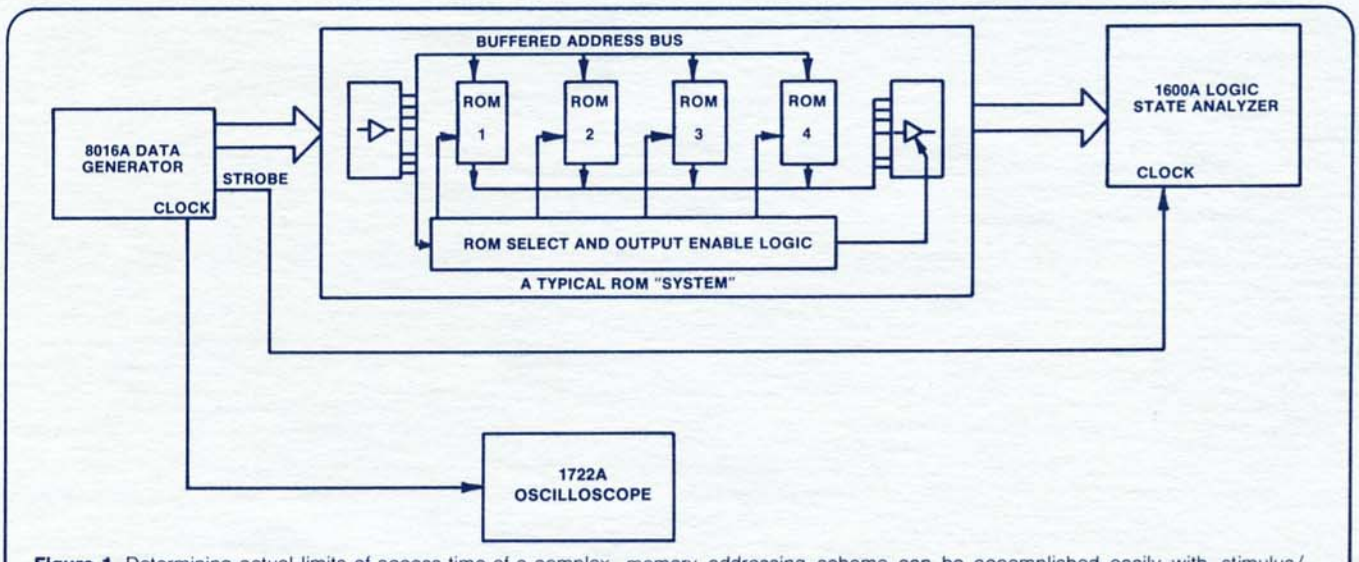


Figure 1. Determining actual limits of access time of a complex memory addressing scheme can be accomplished easily with stimulus/response testing.

For the experiment he set up the word generator to provide an addressing pattern which he considered to be worst case. He then ran this addressing pattern at a moderate speed and stored the data (figure 2) as correctly read by the logic analyzer in the B-memory as a reference. Having stored this data, he used the Exclusive-OR display mode to show if what was currently being read compared exactly to what he stored as a reference. It was then a simple matter to increase the repetition rate of the word generator until the instrument began to detect differences. Then he reduced the frequency just slightly from that point and measured it. The frequency thus read, converted to cycle time, was the actual, not simulated or theoretical cycle time of the memory addressing system.

The previous experiment is representative of a class of problems dealing with real-time data flow. For instance, there are always such questions as how fast can a system respond to certain stimuli, such as interrupts; or how quickly can two keys be struck in sequence on a terminal or calculator keyboard and still have the data read correctly, and so on. Usually, these operations require several levels of logic and most often require an interaction between software and logic. The way to get the most dependable answers to these kinds of questions is through real-time measurement rather than simulation.

Of course, speed is not the only variable that the designer may want to measure on a parametric basis; he might prefer to look at overall system operation. The map display mode (figure 3) of a logic state analyzer allows him to actually watch a program being executed. The designer can establish a correct map pattern by photograph or just in his mind's eye, and it becomes very easy to detect when that pattern changes even on a relatively slight basis. In this way the designer can monitor entire operations while he does things like vary power supply voltage, inject noise, vary temperature or create any sort of environment that he wishes. He can run his prototype or even production units through extensive tests to see at what point the operation begins to "blow up". Not only is it instantly recognizable on the map when a machine begins to do something differently but it is also possible, with the aid of the cursor, to track down the area in which the operation appears to fail first and examine that area in detail with the resultant triggered table display modes.

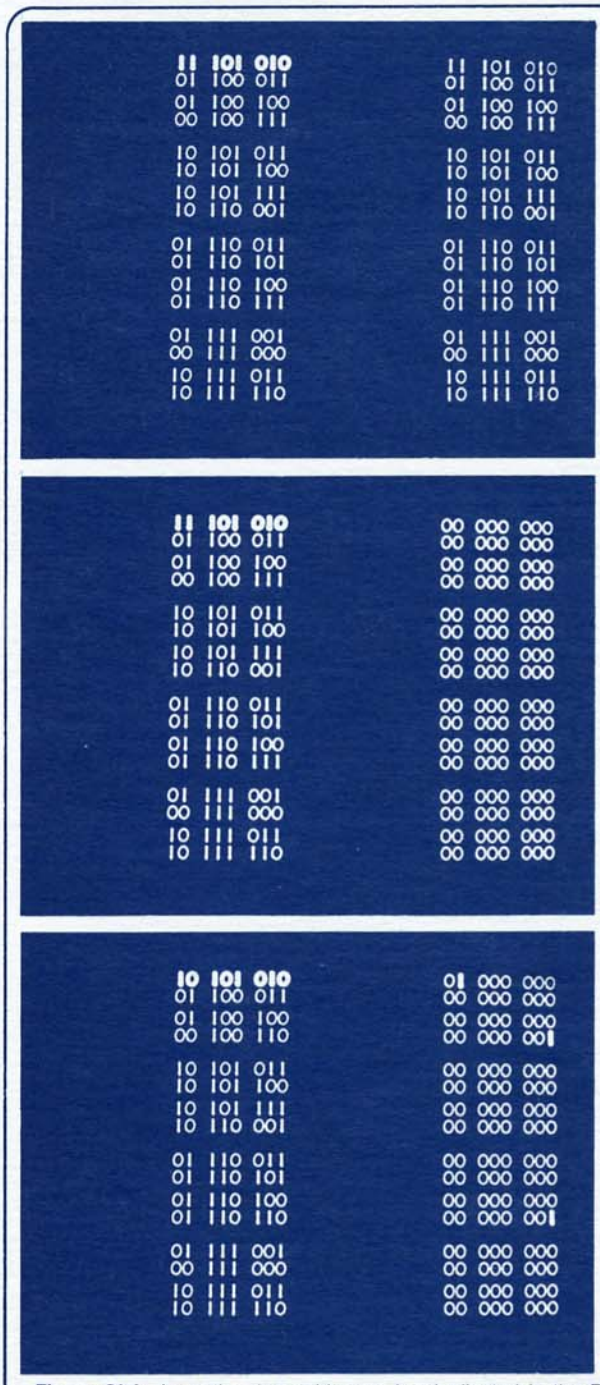


Figure 2(a). An entire data table can be duplicated in the B-memory as a stored reference for comparison with active data. **(b)** The B-memory can be reduced to an Exclusive-OR display of the stored data and the active data being strobed into the A-memory. **(c)** The presence of intensified "1's" in the Exclusive-OR display indicates that logic differences have been detected between the data in the two memories.

Figure 3. The map display mode of a logic state analyzer depicts each 16-bit word as a single dot on screen with a vector drawn between the dots to indicate data sequence. The map allows the designer to view entire operations of his machine watching for pattern changes that indicate the machine is no longer executing the same operations in the same manner.

Figure 4. A 1600S system consists of two logic analyzers, the 1600A and the 1607A, connected together to form one 32-bit analyzer capable of simultaneously displaying and triggering on two independently operating 16-bit data streams. It can easily capture and display simultaneously address and data up to 32-bits in width, show data being read from the same bus at two different threshold settings and Exclusive-OR the display to highlight differences, and show dual-clock systems on one screen for real time analysis of such things as I/O data transfer.

Sensitivity to noise or ringing on bus lines is of considerable concern to the logic designer. Ringing is a common form of "glitch" that often causes erroneous state flow, but to measure threshold sensitivity in a complex system under all sorts of conditions has been a difficult problem. Quite often the worst problems are only present under unique circumstances. E.g., one bit of a 16 bit I/O is read intermittently in error only when nearly all bits in the data switch at once. One method to examine the system's noise sensitivity is to use a 1600S system (figure 4) to independently read the data at two different threshold settings and actively compare the results. The A-memory inputs should read the data at a variable threshold setting under the control of the operator, while comparing each bit (Exclusive-OR display again) with data being read into the B-memory at the correct threshold setting (figure 5a). The entire program can be scanned using the pattern triggers and digital delay generators to provide proper registration. Varying the threshold setting will show which lines have the greatest sensitivity, because differences in the data readings will appear (figure 5b). In this case one bit appears to be sensitive to noise caused by several lines all switching simultaneously. An oscilloscope will offer the additional analysis capability desired once the problem line(s) has been located.

LOST IN SOFTWARE

There is another general class of problems in software based systems that the author sometimes likes to refer to as the "where is it?" type problem. Typically this is the case where the machine is executing software, but it is not executing the software correctly. The machine is stuck off in a loop somewhere and the first problem faced by the designer or troubleshooter is to find out where in the software is it stuck, why is it stuck, and how does he bring it back? This is a relatively simple problem to solve with a logic state analyzer, because he can simply turn off the trigger switches and allow the machine to free run and acquire whatever data is present on the address lines. It is then a simple matter to either map the data, and ascertain in what area of code the machine is operating directly from the coordinates of the map: or, even easier, he can press the single-shot switch and arbitrarily grab sixteen words of address data to compare with his program listing. It is particularly useful to use a 1600S to allow the troubleshooter to see not only the address or program sequences that are being executed, but the associated data or instruction words as they actually appear in response to the addressing. Finding software errors or data dependent errors is much quicker and more accurate, because there is no need to speculate on what sort of instruction might be causing the erroneous state flow; the actual instruction is recorded. Having located where in his code the machine is stuck, the troubleshooter can select meaningful trigger words to either "walk" his way through to the point at which exiting from the loop is expected,

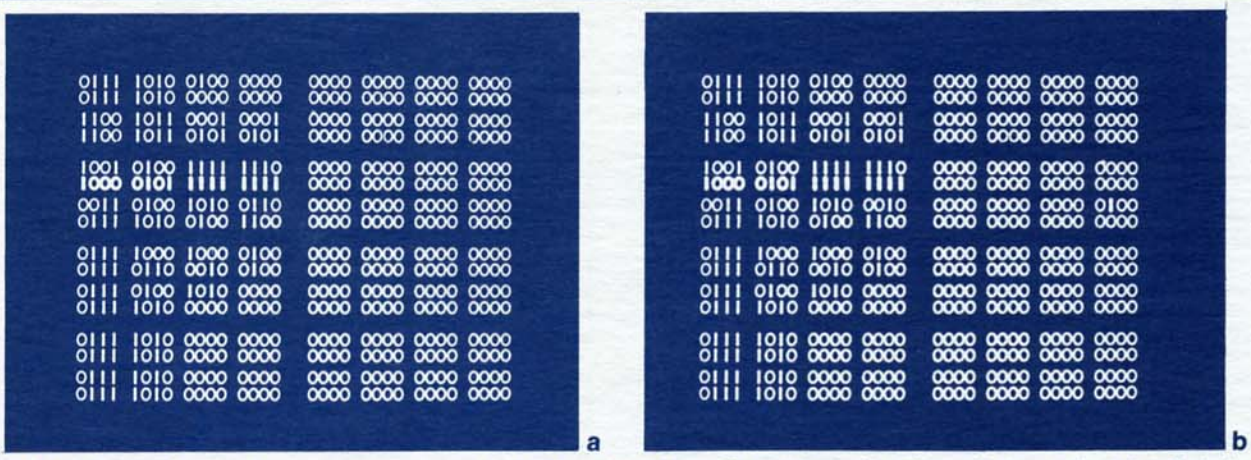


Figure 5(a). A 1600S system is used to measure data on an I/O bus read at two different threshold settings with the resultant data Exclusive-OR'd in the B-memory display position. **(b)** On the word following the highlighted trigger word in the A-memory, one of the bits dipped momentarily to a zero. As the threshold setting was adjusted high enough, the excess ringing caused by the simultaneous switching of a large number of bits on the same clock showed up as a logic error.

or he can use the END display trigger mode to walk backwards to the origin or entry point of the loop if he so desires. Usually finding where the processor came from is more challenging than finding out where it is!

The causes of failure to exit from loops, or causes for arriving in strange areas of code are of course manyfold. Such things as software errors, flag signals that fail to be set or reset, or fail to arrive on time, noise, any number of things can cause the machine to go crashing off into the software "weeds". The reason a logic state analyzer can help solve these problems quickly is that it does not require that the user know anything whatever about the dimensions of the problem to make the first measurement. He simply grabs arbitrarily 16 words of data, or looks at the map and asks "where is it?". The logic analyzer tells him exactly. Solving the problem from that point may or may not be straightforward; it may require thoughtful analysis to determine the cause of the error, but the logic state analyzer will certainly point out the location to look at next for that cause.

THE DATA DOMAIN AND THE TIME DOMAIN ARE OFTEN INSEPARABLE

Another fairly common problem calling for real-time analysis is one mentioned previously, one where a flag signal is being incorrectly read. A generalized example is shown in figure 6. At program address 1002, the external I/O signal must be "set" in order for the machine to correctly exit the loop. An oscilloscope or logic probe verifies that the flag signal is present but the machine fails to recognize it. The problem is one of timing and to analyze it properly the oscilloscope must be synchronized to the proper moment in state time, not triggered from the detection of the signal itself. One of the key features of a logic state analyzer is the trigger output. Two trigger outputs are provided. One, when a preset pattern trigger occurs and is recognized, a second one when the delay is counted down. In this case, the pattern trigger output can be used directly, because it is known exactly at which program address, 1002, the flag signal is to be read. Externally triggering the scope on the occurrence of that address enables examination of the flag signal synchronized in time to the software. Once a state-time reference is established, determining whether it is early or late is easy.

Note the need again for time domain analysis together with state analysis. The two are quite often inseparable and form the basis for instrumentation such as the "gold button" oscilloscope from Hewlett-Packard that provides single button selection of either time domain display or logic state analysis in a system (figure 7).

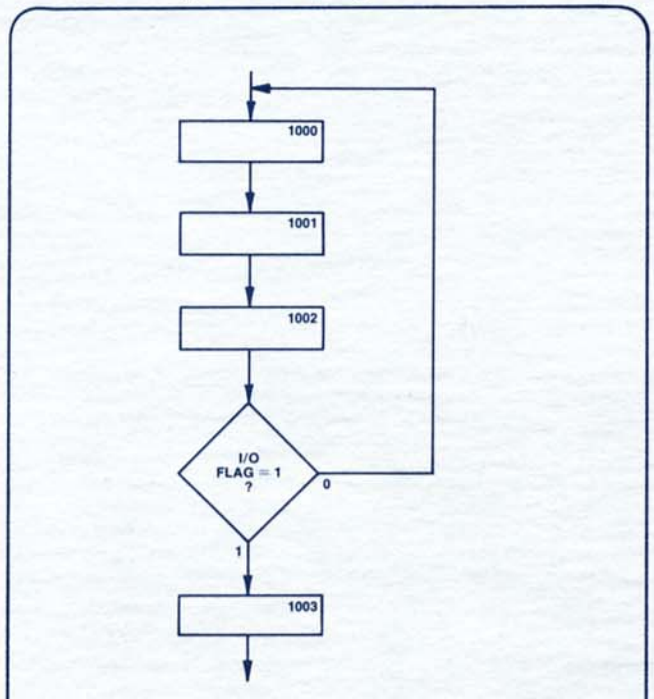


Figure 6. Illustrates a typical loop problem. Failure to exit from the loop is caused by failure to detect or properly read the "flag" signal at program address 1002. An oscilloscope can be used to examine the "flag" line synchronized in time to address 1002 via the trigger output of a logic state analyzer.

Figure 7. The 1740S system consisting of a 1607A logic state analyzer and a specially modified 1740A oscilloscope was created in response to the need to combine both time and data domain measurement capability in a convenient form.

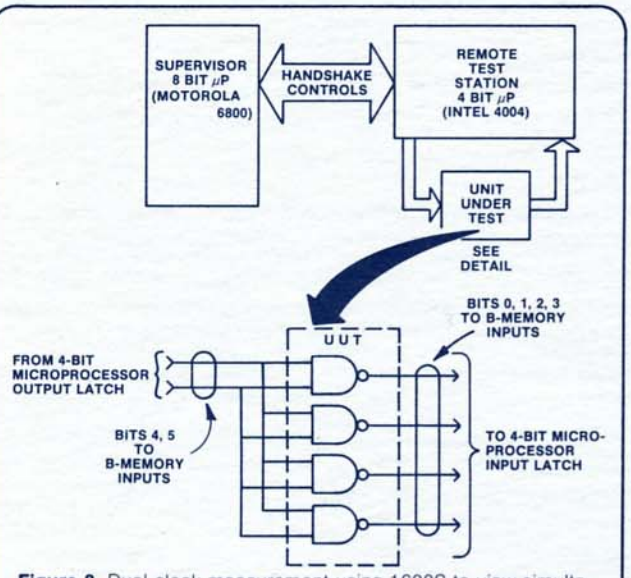


Figure 8. Dual clock measurement using 1600S to view simultaneously on one screen both the software of the microprocessor unit (the program addressed) and the results of the functional test at the I/O of the test station.

REAL TIME DUAL CLOCK ANALYSIS ACROSS AN I/O PORT

As a final example, consider the problems of dual clock troubleshooting across an I/O port. In this example, an I/C tester has been designed which utilizes a 4-bit microprocessor (Intel 4004) to form a test station under handshake control from an 8-bit supervisory microprocessor (Motorola 6800). The supervisor instructs the test station to execute a complete functional check of the truth table of a quad 2-input NAND gate. What should be verified under real-time conditions is that the system operates correctly, that the hardware responds correctly across the I/O as a unique function of the software.

Again a 1600S system is used to provide the capability of reading data on two independent clocks (figure 8). Memory A displays 8 bits of the program sequence plus the 8 bit instructions of the remote unit, and memory B displays the execution of the functional tests by the test station. The two displays are sync'd together (figure 9) at the unique program step, hex address 22E2 via the bus trigger. The B-memory clearly shows the 4 possible input conditions on channels 4 and 5 and the 4 outputs correct for the NAND function read on channels 0 through 3. Note only 4 lines (clocks) of data are read into the B-memory because the test was executed single-shot.

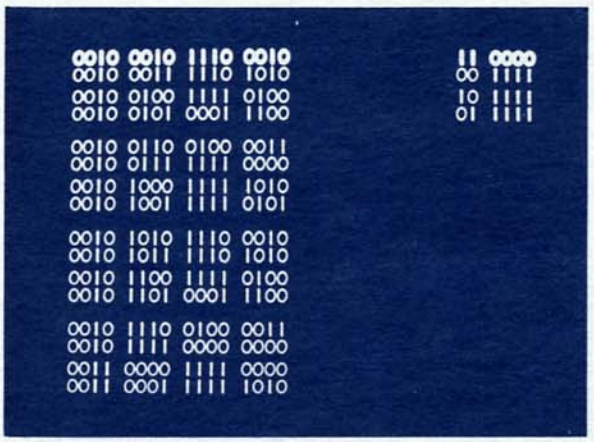


Figure 9. A 1600S in a dual clock measurement. The A-memory is showing the software (8 bits of program count +8 bit instruction words) of the remote microprocessor (Intel 4004) while the B-memory shows the I/O data at the test station testing a quad 2-input NAND gate.

SUMMARY

The last example, one involving data transfer across an I/O, executed in real-time under dual-clock conditions, measured without interference by the measurement tool itself, summarizes in one experiment many of the special abilities of the logic state analyzers. In other examples, more mundane, perhaps, but often troublesome measurements were illustrated involving parametric as well as logic state flow problems. The reader should be able to identify similar problems he has personally encountered. He will make the ultimate judgment of the worth of logic state analyzers in terms of his own potential time saving or increased measurement capability.

REFERENCES

Bowers, Dan M., "Comes der Revolution-Again." Modern Data, January 1975 pp 36-37.

Davidow, William, "The Coming Merger of Hardware and Software Design." Electronics, May 29, 1975 pp 91-94.

Farnbach, William, "Troubleshooting in the Data Domain is Simplified by Logic Analyzers." Electronics, May 15, 1975 pp 103-108. (Reprinted with permission, by Hewlett-Packard as Application Note 167-5.)

Forsberg, R., Neth, J., "Microprocessors & Microcomputers: What will the Future Bring?" EDN, November 20, 1974 pp 24-29.

Garrow, R., Hou, S., Lally, J., Walker, H., "Microcomputer-Development System Achieves Hardware-Software Harmony." Electronics, May 29, 1975 pp 95-102.

House, Charles H., "Engineering in the Data Domain Calls for a New Kind of Digital Instrument." Electronics, May 1, 1975 pp 75-80. (Reprinted with permission, by Hewlett-Packard as Application Note 167-4.)

Morrill, J., Small C., "The Logic State Analyzer, A Viewing Port for the Data Domain." Hewlett-Packard Journal, August, 1975.

Wagner, William, "Unravelling Problems in the Design of Microprocessor Based Systems." Hewlett-Packard Journal, August, 1975.

167 SERIES APPLICATION NOTES: This series of application notes includes general information about digital measurements, specific measurements on various microprocessor systems, and triggering from serial pattern recognition. Contact your HP Field Engineer for specific information about these application notes.

VIDEO TAPE SERIES: This four hour series titled "The Data Domain Its Analysis and Measurements" introduces logic state analysis and measurement techniques unique to the data domain. Contact your HP Field Engineer for price and availability of this color tape series.

5952-1996

PRINTED IN U.S.A.