SLAC–PUB–4766 (REV)
November 1988
(I)

PASHA—AN APPROACH TO COMPUTER-AIDED HARDWARE DEBUGGING*

ROMAIN C. AGOSTINI, HELMUT V. WALZ, DAVID B. GUSTAVSON AND LOY BARKER
Stanford Linear Accelerator Center, Stanford University, Stanford, California 94309

ABSTRACT

An interactive computer-aided diagnostic tool for the SLAC FASTBUS SNOOP has been developed. The distributed software package, written in FORTH and executing concurrently on an IBM PC and on the SNOOP, automatically tests the board and allows pinpointing faults by running a set of specific tests. Special attention has been given to the menu-driven man–machine interface which guides the use through the debugging session. Details of the design and results of field trials are reported.

INTRODUCTION

Intelligent modules residing on a backplane bus often reach a degree of complexity that defies conventional validation, testing and troubleshooting methods. Highly trained technical people have to spend a considerable amount of time setting up test environments and designing elaborate test procedures for dealing with these tasks. Unfortunately, a lot of effort is wasted as many tests are designed on a per module basis, too specific to be reused in a slightly different environment, or so badly documented that it is easier 'reinventing' them again if need should arise. Generally speaking, manual hardware debugging and testing has to deal with the following problems:

- Highly specific and thus non-reusable test procedures
- Wasted time and effort due to multiple designing of identical tests
- Lack of adequate documentation inherent to 'on-the-fly' development
- Even simple functional tests need to be run by specialists
- Incomplete error coverage due to a non-systematic design approach
- Long repair delays

A lot of these problems can be avoided with automated test procedures that are computer-controlled, coded in software and hence only developed once. In such an environment, some kind of computer works its way through a predefined test scheme, outputs test vectors, analyzes response to them and generates a test protocol. The following list tries to summarize the advantages and disadvantages of such an automated test approach:

+ one-time design effort	– designing and coding the test system may be tedious
+ systematic approach guarantees an excellent error coverage	– additional hardware for stimuli needed
+ no supervision of test run required	
+ experts only needed for actual repair	
+ short repair time due to automatic identification of problem area	
+ cost-effective if sufficient modules	

Automated test systems thus seem to offer a multitude of advantages. In order to investigate their real-life implications, SLAC decided to implement such an environment for debugging their SNOOP modules.

THE MODULE UNDER TEST: SNOOP

The SNOOP module, developed at SLAC, is a dedicated Fastbus logic analyzer which resides in a crate and monitors the Fastbus backplane traffic; moreover, it has functional capabilities of acting as a bus slave as well as a bus master. The heart of the module is a Motorola 68000 CPU which interacts with a high-speed ECL section that attaches to and analyzes the bus traffic. Additional information about the hardware structure and its multitude of features can be found in Refs. [1], [2] and [3].

From the software point of view, the SNOOP module is a stand-alone computing system with a multi-tasking operating system supporting the FORTH language. The system software has mainly been written in this language, with time-critical portions having been implemented in assembler. The user interface is a menu-driven Macintosh application (also written in FORTH) which sports user-friendly features such as mouse support, pull-down menus, pop-up dialog boxes, etc. This interface has been described in Ref. [4].

The SNOOP module contains over 200 chips on a 6-layer PC board, so it is easy to understand that troubleshooting can rapidly become complicated. Not only does a repair person need to understand the underlying electrical and logical intricacies of its design, he must also have a great deal of knowledge about the software interactions. Moreover, generating external stimuli on the Fastbus as test vectors requires proficiency with the IORFI command interface FBDOS32, (Ref. [5]). Adding all this up shows that a considerable amount of expertise is required for performing even the simplest of checkouts.

THE TEST SYSTEM: PASHA

This led to the idea of implementing an automated test system, capable of performing module checkouts without requiring any user input except for starting the task. Such a system would be of considerable help, both for finding out whether a used module is still 100% functional and for validating newly produced units. In the course of time, more desiderata appeared: besides running in a totally automatic mode in which test vectors are generated by the system, a manual mode where the user can specify test vectors of his own choice should also be supported; the option for a printed test protocol should be available so that a whole test sequence can be run without anybody having to attend (ideal for overnight testing); a loop mode in which the same test is repeated over and over for debugging with a scope or logic probe should be supported, etc.

All these features have been combined into the PASHA test suite – **P**ackage for **A**utomated **S**noop **H**ardware **A**nalysis. The actual test environment (Fig. 1) consists of the SNOOP module under test, an IORFI module for controlling the bus (both modules reside in a Fastbus crate) and an IBM PC/XT. The PC is used as a hardware interface to the IORFI (driver cards connected to the PC bus) and as an operator interface for PASHA (connected via a RS-232C interface to the SNOOP module).

PASHA is a distributed software package which runs concurrently on the SNOOP under test and on the PC. As it is executing on two different platforms, it had to be written in

* Work supported by the Department of Energy, contract DE-AC03-76SF00515.

Poster paper presented at the Nuclear Science Symposium, Orlando, Florida, November 9–11, 1988.

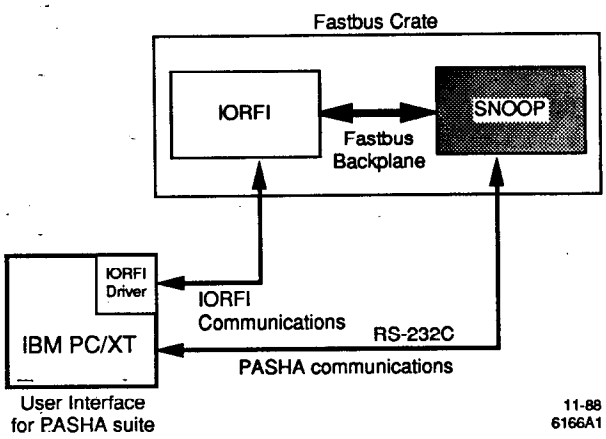

Fig. 1. Test setup.

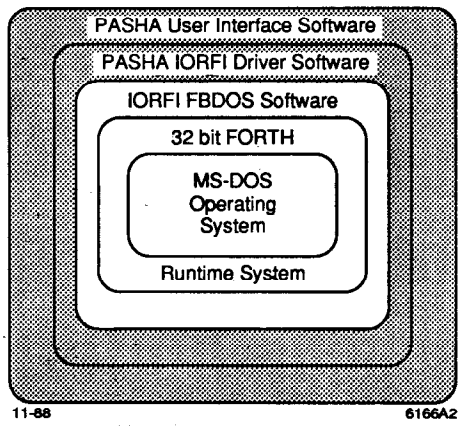

Fig. 2. PC software structure.

two different versions of FORTH (the PC version supports 32 bit constructs implicitly, but the SNOOP version doesn't). Figures 2 and 3 show the software layering on the two machines.

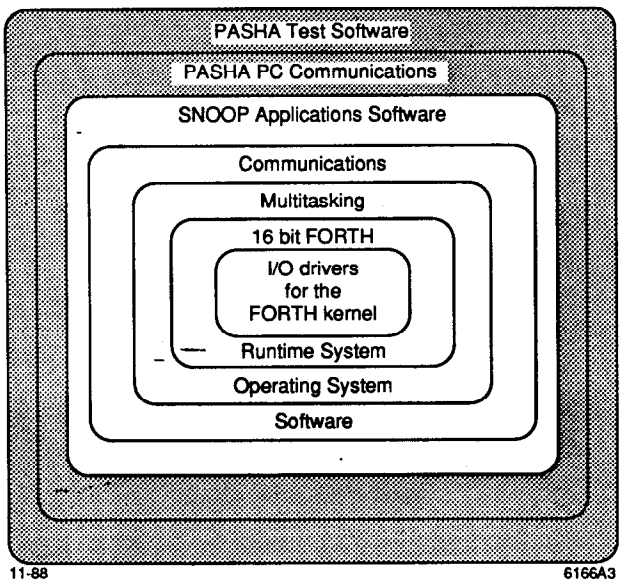

Fig. 3. SNOOP software structure.

The actual test core of PASHA is executing on the SNOOP. This was a convenience choice, as the SNOOP command interface (part of the SNOOP applications software in the figure) can be directly accessed from there, and mechanisms for sending commands to the PC (and thus also to the IORFI) are readily available. This way, all the 'intelligent' processing is done in the 68000 CPU, an the PC is only used as a Fastbus driver (via IORFI) and an operator interface. This is fine, as these functions represent different logical entities and should be kept apart in the software structure as well.

Up to the present date, a dozen separate tests have been designed and integrated into the test suite. In order to keep them independent of each other, they have been implemented as virtually stand-alone modules which build on a common base of variables, data structures and FORTH definitions. This approach guarantees a minimum of side effects and allows for easy servicing and upgrading of specific modules. Clearly, an optimally compact code is neither achieved nor is it aspired to do so, as the modules can be stored on the PC's hard disk and loaded into the SNOOP RAM memory when needed. This way, the growth of the test suite is not impaired by system memory constraints.

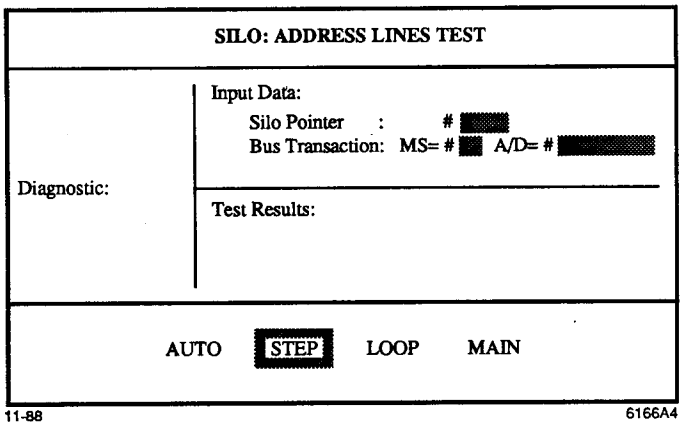

Fig. 4. Menu before test execution.

As mentioned before, the user interface is implemented on the PC. Designed to free the operator from having to memorize exotic command sequences and from learning the FORTH programming language, this interface is both interactive and menu-driven. Upon selection of a specific test, a menu with all pertinent information is displayed (Fig. 4); current choices are highlighted in reverse video. The bottom part of the screen allows specifying the test mode, with the following options available:

- AUTO: The system generates a comprehensive set of test vectors and utilizes them without requesting any input from the operator. These vectors are displayed in the upper right part of the menu, as well as a general diagnostic message (pass or fail) on the left side and more specific information in the lower right part.
- STEP: A single test vector, which has to be specified by the operator, is executed. Upon invoking this choice, the upper right part of the screen displays an input mask and the cursor positions itself on the first field. Illegal input characters are automatically rejected by the system. Results are displayed as in the automatic mode.
- LOOP: Like STEP, but the same test is repeated over and over again until stopped by the user. This allows for investigating the hardware with an oscilloscope or a logic probe.
- MAIN: Displays the main menu where the user can select a specific test or initiate the execution of the whole PASHA suite.

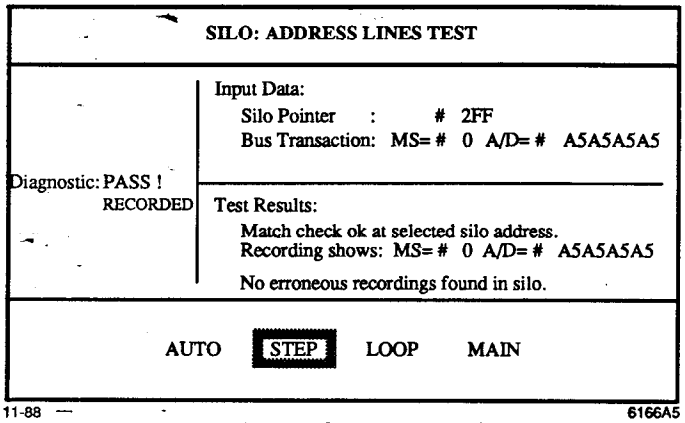

Fig. 5. Menu after test execution.

Figure 5 shows the same menu after a user-specific test vector has been executed and the results analyzed. In this case, the user specified that the SNOOP module should select location #2FF in the silo memory as the next recording location, and specified an address cycle to address #A5A5A5A5 on the Fastbus via the IORFI. The PASHA system subsequently erases the whole silo, loads the silo pointer with the value #2FF, arms the silo for recording any upcoming Fastbus traffic and requests an address cycle to address #A5A5A5A5 from the IORFI. After this has happened, it checks whether the transaction has been correctly recorded at the desired location, makes sure that the transaction doesn't show up at any other silo location (which would indicate shorted address lines, the purpose of this test) and displays the results of its checking. Without the PASHA system, this rather straightforward procedure would require the explicit knowledge of some 20 commands and of a programming language in order to loop through the silo memory for checking out its contents. None of this is required when using PASHA. Moreover, in automatic mode, the user doesn't even have to come up with a test vector, as a complete set is automatically generated.

In more complicated menus, the operator has an additional choice of different submodes which are directly related to the nature of the subcircuit under test. Figure 6 shows a menu where the same basic SNOOP function (generating a Fastbus wait signal upon recognition of a trap) has to be verified in different modes (a single address can be the trap, or two addresses, or an address cycle followed by a data cycle). These submodes are shown in the upper left corner and are selected with the PC's function keys. Choices are again highlighted in reverse video. For the rest, the test works as outlined above.

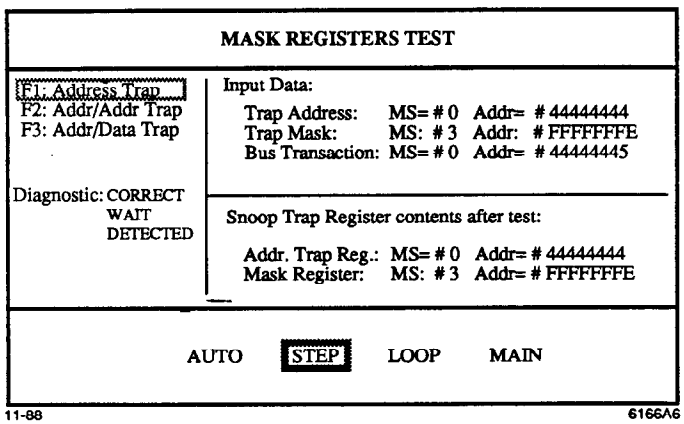

Fig. 6. Menu screen for mask registers test.

In order to validate a new SNOOP module which has just been assembled, a user plugs it into the Fastbus crate with the IORFI module, calls up the PASHA test suite and instructs it to step through all test modules in automatic mode. Upon detection of an error, an acoustic signal is generated and the test procedure is halted; a keystroke from the operator resumes checking. Although this operating mode requires an operator, the test suite can also be run in a completely unattended mode. In order to do this, the user only has to request a printed test protocol, and every test will be executed with appropriate printer output and no halts after errors. Thus, a module can easily be checked overnight. Another advantage of this is that no highly skilled technician is needed just for finding out whether a module is operational (but notice that this is not necessarily true for actually repairing a broken module).

The reader may have noticed that this way of testing requires a fully operational processor part of the module in order to run the operating system, the FORTH runtime system and the PASHA suite. But hardware failures may also occur in this part and prevent the whole system from running. For this purpose, three tests have been written in 68000 assembly language and programmed into EPROMs. These EPROMs are plugged into the sockets which will later host the system software, and the CPU starts executing the test program when booted.

The first EPROM test is a simple counting loop for testing the CPU, the address lines to the memory and the generation of control signals. The second EPROM test reads keys from the IBM PC, echoes them back to the PC's CRT and displays their hex value on the SNOOP front panel LEDs. This way, the communications chip and the RS-232C interface to the PC is checked out. A third EPROM test deals with a VLSI timer/counter chip and interrupts. If these three tests are successfully passed, the system software can be run.

If the test protocol reveals some bugs, the operator can invoke the specific test module from the main menu and run it again, if need be. In order to zero in on the bug, specific test vectors may be entered in the STEP mode and the responses to them analyzed; in the LOOP mode, some hardware probing can be done. The repair person taking over at this time needs to have a more thorough understanding of the SNOOP's inner structure, as he will be using the schematics as well; some knowledge of the SNOOP commands and FBDOS32 is helpful as well. The main point is that up to this point, no specific knowledge whatsoever has been required.

WORK EXPERIENCE WITH PASHA

The present PASHA suite contains 12 tests which deal with the following SNOOP subcircuits:

- Wait generation circuits
- Trap registers
- Mask registers for trap operations
- Silo write enable logic
- Silo counter and pointer handling
- Silo address lines
- Recording start on trap
- Recording stop on trap
- Parity checking circuits
- Fastbus protocol violation tags
- Control and status registers
- Geographical addressing circuits

The test suite has been utilized for complete debugging two SNOOP modules and is now used for doing some checking on two additional modules. As expected, the first debugging sessions pointed out some weak points in the software which have been corrected in an iterative process. The present version has been fairly stable, with some enhancements having been added for convenience.

Running the tests in automatic mode proved to be extremely easy, and a subsequent analysis of the test protocol pinpointed the areas to concentrate on. Sometimes it was even possible to determine the exact cause of errors just by carefully evaluating the protocol. In most cases, however, direct hardware probing was necessary, and at this point both the STEP and the LOOP modes proofed handy. Often, a logic probe was enough for determining the cause of the failure. On a few occasions, it was necessary to use an oscilloscope and/or write some specific command sequences for getting both the Fastbus and the SNOOP into the required state. As mentioned before, the actual repairing does need a person capable of reading a circuit diagram and familiar with digital electronics.

On the whole, the PASHA test suite proved to be very helpful and effective. The nicest feature is that the tests are very comprehensive, which provides good assurance to believe that a module is fully functional after it has been validated by the suite. The excellent error coverage is best documented by the fact that bugs in supposedly working modules were found; they hadn't been detected by manual testing as they only showed up in cases not often encountered during normal operation. And, last but not least, speedy repairing of broken modules has been greatly facilitated by the suite.

FUTURE ENHANCEMENTS

In addition to the development of new test modules, a few software changes could be made to improve the user-friendliness of the suite. One of them would be the implementation of online help menus associated with each test module. Here, the operator could see a brief description of how the test is actually working, what circuits of the SNOOP are tested, and what exact command sequence is executed in the LOOP mode, as this knowledge is required for doing any intelligent probing.

Furthermore, instead of sending a full test protocol to the printer, which tends to be rather lengthy, errors could be logged selectively in order to speed up the process. Writing to a disk file is also potentially useful.

SUMMARY

Emerging needs for computer-controlled hardware testing have been discussed, and the implementation of such a test suite at SLAC has been described. Positive results for both increased error coverage and decreased trouble-shooting time show that automated hardware debugging is the correct approach to the problem of increasing system complexity.

REFERENCES

[1] H. V. Walz, D. B. Gustavson and R. Downing, *Progress on the SLAC SNOOP Diagnostic Module for FASTBUS*, IEEE Transactions on Nuclear Science, NS–30 **1**, pp.220–222 (1983).

[2] H. V. Walz and D. B. Gustavson, *Status of the SLAC SNOOP Diagnostic Module for FASTBUS*, IEEE Transactions on Nuclear Science, NS–30 **4**, pp. 2276–2278 (1983).

[3] D. B. Gustavson and H. V. Walz, *SLAC FASTBUS Snoop Module — Test Results and Support Software*, IEEE Transactions on Nuclear Science, NS–33, **1**, pp. 811–813 (1986).

[4] D. M. Gelphman, D. B. Gustavson and H. V. Walz, *An Interactive Interface for Fastbus Diagnostics*, IEEE Transactions on Nuclear Science, NS–35, **1**, pp. 303–305 (1988).

[5] C. Logg, *FASTBUS Diagnostic Operating Interface (FBDOS)*, Stanford Linear Accelerator Center, August 1982.