1 **IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

2

3

4 In re Application of Jed Margolin

5 Serial No.: 11/130,939 Examiner: Phung M. Chung

6 Filed: 05/17/2005 Art Unit: 2117

7 For: MEMORY WITH INTEGRATED PROGRAMMABLE CONTROLLER

8

9 Mail Stop Amendment
10 Commissioner for Patents
11 P.O. Box 1450
12 Alexandria, VA 22313-1450
13

14 **AMENDMENTS AND RESPONSE**

15

16 Dear Sir:

17

18 In response to the Office Action mailed June 25, 2007, please enter the following
19 amendments and consider the following remarks.

20

21 Amendments to the claims begin on page 2 of this response. Remarks begin on page 8 of
22 this response.

23

24

Claim Amendments

Please cancel claims 4, 7, 9, 12, and 13 without prejudice.

Claims 1, 2, 3, 5, 6, 8, 10, 11, 14, 15, and 16 are amended as follows. No new matter is added as a result of the claim amendments.

Claim 1. (currently amended) A single chip memory comprising:

(a) a memory array;

(b) a processor;

(c) a processor RAM memory;

(d) a multiplexor;

~~whereas said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, and a user's system.~~

whereas:

(a) said processor is connected to said processor RAM memory and said multiplexor;

(b) said memory array is also connected to said multiplexor;

(c) said memory array is a read/write memory;

whereby:

(a) said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, and a user's system;

(b) said user's system uses said multiplexor to store a program into said processor RAM memory;

(c) said processor uses said program in said processor RAM memory to test said memory array; and

whereas said program is an algorithmic test program.

Claim 2. (currently amended) The single chip memory of claim 1 further comprising a non-volatile memory connected to said processor, said processor RAM memory, and said mulitiplexor.

Claim 3. (currently amended) The single chip memory of claim 1 further comprising a programmable clock connected to said processor.

Claim 4. (canceled)

Claim 5. (currently amended) The single chip memory of claim 1 whereby said ~~multiplexor is used to load said processor RAM memory with a~~ program is also used by said processor to perform one or more functions selected from a group comprising data pattern matching, moving data, graphics primitives, data encryption, and data decryption.

Claim 6. (currently amended) A single chip memory comprising:

(a) a memory array;

(b) a processor;

(c) a processor RAM memory;

(d) a multiplexor;

(e) a non-volatile memory;

~~whereas said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, and a user's system; and~~

~~whereby said multiplexor is used to load said processor RAM memory with a program used by said processor.~~

whereas:

(a) said processor is connected to said processor RAM memory, said multiplexor, and said non-volatile memory;

(b) said memory array is also connected to said multiplexor;

(c) said memory array is a read/write memory;

whereby:

(a) said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, said non-volatile memory, and a user's system;

(b) said user's system uses said multiplexor to store a program into said processor RAM memory;

(c) said processor uses said program in said processor RAM memory to test said memory array;

(d) said processor uses said non-volatile memory to store the results of said program in said processor RAM memory used to test said memory array; and

whereas said program is an algorithmic test program.

Claim 7. (canceled)

Claim 8. (currently amended) The single chip memory of claim 6 further comprising a programmable clock connected to said processor.

Claim 9. (canceled)

Claim 10. (currently amended) The single chip memory of claim 6 whereby said program is also used by said processor ~~used~~ to perform one or more functions selected from a group comprising data pattern matching, moving data, graphics primitives, data encryption, and data decryption.

Claim 11. (currently amended) A single chip memory comprising:

(a) a memory array;

(b) a processor;

(c) a processor RAM memory;

(d) a multiplexor;

(e) a non-volatile memory;

(f) a programmable clock;

~~whereas said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, and a user's system; and~~

~~whereby said multiplexor is used to load said processor RAM memory with a program used by said processor to test said memory array.~~

whereas:

(a) said processor is connected to said processor RAM memory, said multiplexor, and said non-volatile memory;

(b) said memory array is also connected to said multiplexor;

(c) said memory array is a read/write memory;

(d) said programmable clock is connected to said processor;

1 whereby:

2 (a) said multiplexor controls and arbitrates access between said memory array, said

3 processor, said processor RAM memory, said non-volatile memory, and a user's system;

4 (b) said user’s system uses said multiplexor to store a program into said processor RAM

5 memory;

6 (c) said processor uses said program in said processor RAM memory to test said memory

7 array;

8 (d) said processor uses said non-volatile memory to store the results of said program in said

9 processor RAM memory used to test said memory array; and

10

11 whereas said program is an algorithmic test program.

12

13 Claim12. (canceled)

14

15 Claim13. (canceled)

16

17 Claim 14. (currently amended) A method for providing a self-testing single chip memory

18 comprising the steps of:

19 (a) providing a memory array;

20 (b) providing a processor;

21 (c) providing a processor RAM memory;

22 (d) providing a multiplexor;

23

24 ~~whereas said multiplexor controls and arbitrates access between said memory array, said~~

25 ~~processor, said processor RAM memory, and a user's system; and~~

26

27 ~~whereby said multiplexor is used to load said processor RAM memory with a program used~~

28 ~~by said processor to test said memory array.~~

29

30 whereas:

31 (a) said processor is connected to said processor RAM memory and said multiplexor;

32 (b) said memory array is also connected to said multiplexor;

(c) said memory array is a read/write memory;

whereby:

(a) said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, and a user's system;

(b) said user's system uses said multiplexor to store a program into said processor RAM memory;

(c) said processor uses said program in said processor RAM memory to test said memory array; and

whereas said program is an algorithmic test program.

Claim 15. (currently amended) The method for providing a self-testing single chip memory of claim 14 further comprising the step of providing a non-volatile memory connected to said processor, said processor RAM memory, and said multiplexor.

Claim 16. (currently amended) The method for providing a self-testing single chip memory of claim 14 further comprising the step of providing a programmable clock connected to said processor.

1 **REMARKS**

2

3 Section 1. Prior Art cited against Applicant

4

5 Before addressing the Examiner's specific rejections it will be useful to discuss the prior art cited
6 against the Applicant.

7

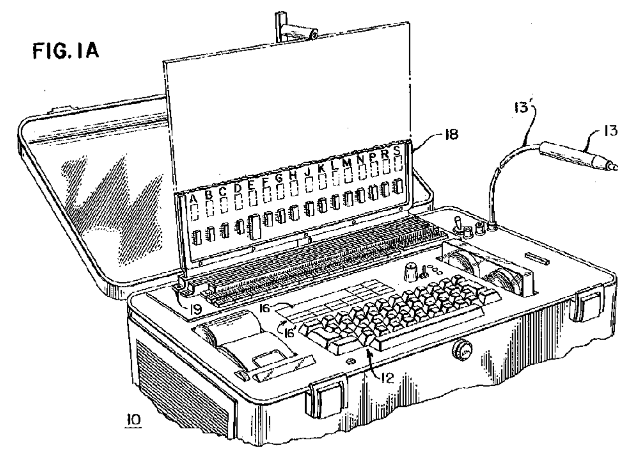
8 **1.** U.S. Patent 4,194,113 **Method and Apparatus For Isolating Faults in a Logic Circuitry**,
9 issued March 18, 1980 to Fulks et al. ("Fulks").

10

11 From Column 6, lines 3 – 6:

12 The digital tester of the invention, hereinafter referred to as the "PSP" (portable service
13 processor), is a processor-oriented portable tester especially suited to testing printed circuit
14 boards. It is a digital logic circuit tester that can detect and isolate faults on digital printed
15 circuit boards.
16
17 From Column 6, lines 29 – 38:
18
19 The complete PSP unit 10 is shown in FIG. 1A. A printed circuit board 18 being tested by
20 the PSP is connected into edge connector 19 (printed circuit boards under test are referred to
21 hereinafter as "boards under test".) It should be noted that various devices in board 18 are
22 labeled by letters; such labeling permits the PSP to instruct the operator as to which nodes to
23 probe during the fault isolation procedure, subsequently described in detail. Keyboard 12
24 permits the operator to enter data and commands into the PSP.
25
26 From Column 22, lines 40 – 49:
27
28 If the board under test does not pass the signature analysis test at each output node, the PSP
29 then goes into an "automatic guided probe" signature analysis testing procedure wherein the
30 display unit informs the operator, on a node by node basis, which node to probe next. The
31 various nodes of the board under test or product under test are identified by any convenient
32 means, such as labeling the components by letters, as shown in FIG. 1A, so that such nodes
33 are readily recognizable by the operator.
34
35
36 Figure 1A, reproduced below, shows the printed circuit board to be tested (18) plugged into the
37 tester.

38



The Fulks patent is for a computer-controlled digital signature analyzer used to isolate faults on digital printed circuit boards. The computer stores the signatures and other information about the board being tested and starts by performing a go/no-go test through the board's edge connector. If the board fails, the computer directs the operator where to place a probe to read the signatures of particular circuit nodes. It then compares the signature of the nodes to the signature of the same nodes obtained from a known-good printed circuit board.

Fulks uses a separate high-speed processor (HSP 29 in Figure 2) to interface to the printed circuit board's connector. This is succinctly explained in application 05/895,898 (now U.S. Patent 4,196,386 **Method and portable apparatus for testing digital printed circuit boards** issued April 1, 1980 to Phelps) incorporated by reference in Fulks. In Phelps Column 9, lines 44 – 58:

> According to the present invention, a high speed processor (HSP), which is programmable to operate at many times the speed of the comparatively slow main processor is utilized to operate in response to and simultaneously with the main processor to sequentially route data received in parallel format from the main processor to predetermined ones of the 192 driver/sensor circuits. The fast sequential outputting is accomplished by means of a subroutine, referred to herein as an "H" file, in the form of object code for the HSP and stored in the HSP memory. This subroutine defines groups of specific pins of the board under test as "destination busses" and sequentially routes sixteen bit data words each contained in a respective single instruction in the test program to such "destination busses".

In Fulks, we find out that the HSP has a programmable clock. In Column 14, lines 53 – 68:

> Still referring to FIG. 5, clock circuit 153 includes a number of conventional counters, registers, flip flops, and some control gating circuitry to produce high speed programmable clock signals utilized to control the operation of the HSP. The range of the cycle times of the programmable clock signals is from 150 nanoseconds to approximately 12.5 microseconds, in 50 nanosecond increments. Clock circuit 153 includes a register which is loaded from

> main bus 27 to determine the programmable cycle time of the HSP. The cycle time of the HSP controls the rate at which PSP 10 switches from one input/output pin to another during the testing of the board under test. Clock circuit 153 may be readily implemented utilizing Texas Instruments 74LS175 latches, 74S74 flip flops, and 74S161 counters as the main components thereof.

The central part of Fulks' invention uses signature analysis. One of the shortcomings of signature analysis is that feedback loops must be broken. From Column 25, lines 16 – 32:

> The signature analysis techniques described above "break down" if there is a feedback loop in the digital logic circuitry of the board under test. This is because an error in the response at any node in a digital feedback loop ordinarily propagates very quickly around the loop to produce faulty responses at every node in the loop, making ineffective the simple procedure of tracing along the topology pattern of a device under test until a device is found with a faulty output and all good input. As previously mentioned, all signature analysis techniques have been unsuccessful in isolating faults at nodes which occur within loops and digital circuits. Until the present invention, only expensive "factory testers" which store the entire data stream at every node, and which therefore require very large memory storage capability, have been able to isolate faults within feedback loops in digital circuits.

Fulks solves this problem by storing a smaller data stream. From Column 25, lines 33 – 43:

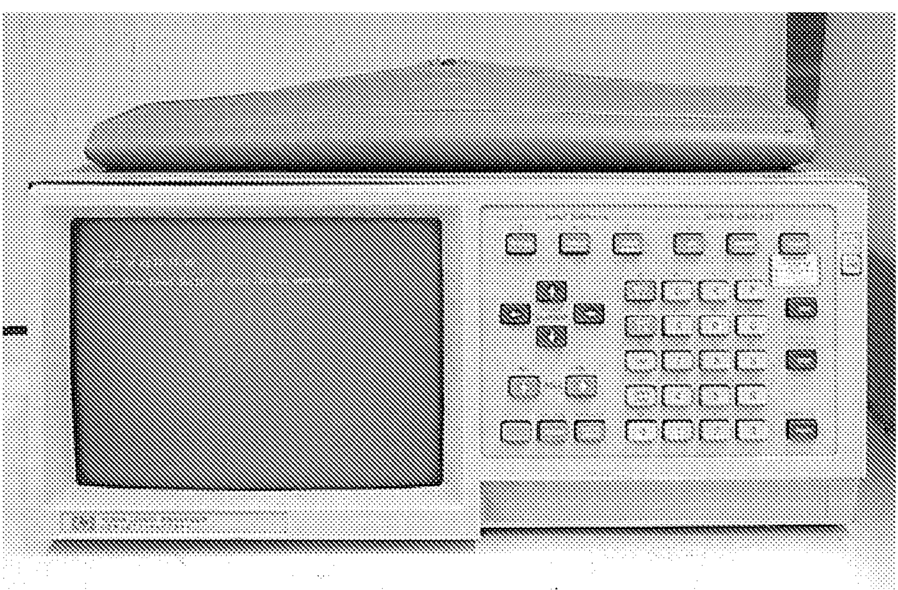
> According to the present invention, a new procedure has been discovered which isolates faults within loops in digital circuits by determining and storing the initial state of each node in the loop and determining and storing the time at which each node in the loop initially failed. The inventive procedure has been found to compare such stored information for boards under test (with digital feedback loops therein) with data from known good boards of the same type to isolate (with a very surprisingly high degree of success) the defective components of the board under test.

Because signature analysis is at the heart of Fulks' invention it is necessary to discuss what signature analysis is.

Signature analysis was developed several decades ago to troubleshoot printed circuit boards containing clocked logic.

Before that, the primary instrument used to troubleshoot printed circuit boards was the oscilloscope. The oscilloscope probe was connected to the various nodes and the user determined whether the signal was correct. For SSI logic this was done by first looking at the inputs and then looking at the output(s). This required knowledge of the SSI part. For larger circuits this required knowledge of how the circuit was supposed to work. The job was made somewhat easier by the

1 development of triggered oscilloscopes with two or more channels and an adjustable delayed
2 sweep.
3
4 Troubleshooting a logic board with an oscilloscope was frequently a painstaking, time-
5 consuming process.
6
7 The alternative was known as shot-gunning, where integrated circuits suspected of being bad
8 were replaced, sometimes on an educated guess, sometimes at random.
9
10 Logic Analyzers were available, which allowed a large number of digitals signals to be captured
11 and displayed.
12
13 From Wikipedia (**http://en.wikipedia.org/wiki/Logic_analyzer**):
14 A **logic analyzer** displays signals in a digital circuit that are too fast to be observed by a
15 human being and presents it to a user so that the user can more easily check correct
16 operation of the digital system. Logic analyzers are typically used for capturing data in
17 systems that have too many channels to be examined with an oscilloscope. Software running
18 on the logic analyzer can convert the captured data into timing diagrams, protocol decodes,
19 state machine traces, assembly language, or correlate assembly with source-level software.
20
21 Here is a picture of an HP 1630A Logic Analyzer.
22



23
24 *The 1630A Logic Analyzer combines four logic analysis functions in one benchtop*
25 *instrument providing the versatility required in up to 35 bits and timing up to 8*
26 *bits. Features timing analysis at 100 MHz to check hardware and status signals;*
27 *state analysis at 25 MHz to trace program and software flow; performance*

analysis to optimize code; interactive state/timing analysis to integrate circuits and code. The 1630A can become your single most important tool for logic design, development, and testing. The 1630A has 35 input lines, of which 8 lines may be used for timing analysis. Standard, built-in HP-IB and HP-IL interfaces.

The signals could be displayed as digital waveforms, as binary, or as hexadecimal. The analyzer could perform a large number of functions too numerous to list.

Using a Logic Analyzer required considerable experience. Logic Analyzers were also expensive, which made them rare, especially in the field, which made Signature Analysis more attractive..

An early patent for a Logic Analyzer is U.S. Patent 4,040,025 **Logic state analyzer** issued August 2, 1977 to Morrill, Jr., et al. (Assignee is The Hewlett Packard Company).

Signature Analysis was developed to allow trouble shooting to be done in the field with an inexpensive instrument.

An early patent for Signature Analysis is U.S. Patent 3,976,864 **Apparatus and method for testing digital circuits** issued August 24, 1976 to Gordon, et al. . (Assignee is The Hewlett Packard Company).

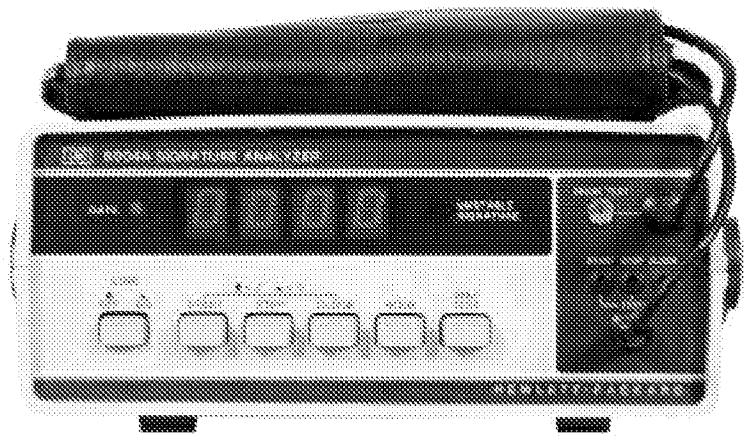
From the Abstract:

> A device is disclosed which converts a digital signal or bit stream into a digital signature repesentative of the digital signal by means of a feedback shift register. The apparatus may be used to identify and characterize digital signals at various test points in an apparatus for testing purposes. Signatures for digital signals from properly operating circuits can be recorded in a variety of fashions for later comparison with signatures of digital signals from circuits under test. The comparison of the signatures enables a person using the apparatus to determine whether the circuit under test is operating properly and, if it is not, to locate the fault in many instances. The apparatus may also be used to examine digital signals to enable identification of transient errors.

In Signature Analysis the data stream from a circuit node is clocked into a Linear-Feedback Shift Register (LFR) during a specified time period selected by a control program. After the time period is over, the contents of the LFR are read out and usually displayed as four non-standard hexadecimal characters called the Signature. This signature is compared to the signature

1 produced by known-good hardware. The Signature Analyzer uses only four signals: Clock, Start,
2 Stop, and Data (from the node to be tested).

3 By using a LFSR, discrepancies in the data stream have a high probability of being detected
4 regardless of when they occur or how many occur. See Appendix A, the ***Hewlett-Packard***
5 ***Journal*** for May 1977 for a detailed description of Signature Analysis and the HP 5004A, which
6 is pictured below.



7
8 *HP5004A*
9
10

11 There are a number of limitations of Signature Analysis which are the tradeoffs for the simplicity
12 of the method which makes it easy to use.

13
14 1. In its pure form, feedback loops must be broken. (Fulks adds memory to record the data
15 streams which adds to the cost and complexity of the instrument, both of which are contrary to
16 the spirit of Signature Analysis.)

17

18 Breaking a feedback loop generally requires adding a gate to a circuit which usually increases
19 the cost directly or indirectly because of the traces which must be added to the printed circuit
20 board. Adding a gate also increases the propagation time of the circuit, which can reduce the
21 system's performance.

22

2. Signature Analysis is only useful for testing circuits that produce a data stream, which makes it useless for testing RAM. RAM must be tested by using a variety of data patterns. As taught in the application by the present Applicant, this is done algorithmically by following a number of steps. See Table 1(a) through Table 1(k) in the present application.

Note that although Signature Analysis can be used to test ROM in a CPU-controlled system it is more efficient for the CPU to read the Read-Only Memory locations and perform a Cyclic-Redundancy Check (CRC). This produces a checksum which can be compared to the checksum of the file data used to program the ROM. It is faster than placing a probe on each data line and it doesn't require known-good hardware.

One of the products that used Signature Analysis was the coin-operated video game ***Battlezone*** by Atari, Inc. produced in 1980.

See Appendix B for background on the game, the Signature Analysis instructions on the schematics, and sections of the schematics showing some of the signatures.

Battlezone was one of the few games produced by Atari that used Signature Analysis. It wasn't worth the cost. Future games used more-robust hardware design and more-comprehensive Built-In Self-Test and Diagnostic software.

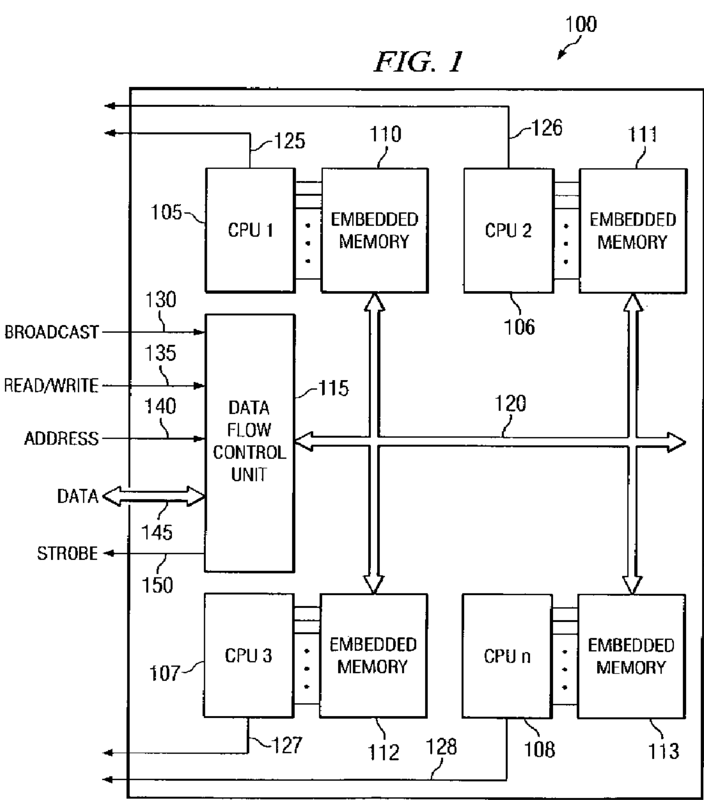
It should be noted that the present Applicant, Jed Margolin, worked for Atari, Inc. and Atari Games Corporation (one of Atari Inc.'s successor companies) from 1979 – 1992.

Margolin was the Hardware Engineer for the Battlezone project. This makes Margolin a Person Having Greater-than-Ordinary Skill in this Art.

It should also be noted that Margolin is the inventor listed on U.S. Patent 4,195,293 **Random dot generator for raster scan video displays** issued March 25, 1980. The heart of the invention is a Linear-Feedback Shift Register. This makes Margolin a Person Having Greater-than-Ordinary Skill in this Art, too.

1 **2.** U.S. Patent 7,155,637 **Method and apparatus for testing embedded memory on devices**
2 **with multiple processor cores** issued December 26, 2006 to Jarboe, Jr. et al. ("Jarboe").
3
4 From the Abstract:

5 The disclosed method and apparatus enables the testing of multiple embedded memory
6 arrays associated with multiple processor cores on a single computer chip. According to one
7 aspect, the disclosed method and apparatus identifies certain rows and columns within each
8 of the embedded memory arrays that need to be disabled and also identifies certain
9 redundant rows and columns in the embedded memory array to be activated. According to
10 another aspect, the disclosed method and apparatus generates a map indicating where each
11 of the memory failures occurs in each embedded memory array. If the testing process
12 determines that the embedded memory array cannot be repaired, then a signal is provided
13 directly to an external testing device indicating that the embedded memory array is non-
14 repairable. Similarly, if the testing process determines that the failures in the embedded
15 memory array can be repaired, then a signal is provided directly to an external testing
16 apparatus indicating that the embedded memory array is repairable. Lastly, if no failures are
17 found in an embedded memory array, then a signal is provided to an external testing
18 apparatus indicating that the embedded memory array contains no failures.
19
20 Here is Jarboe Figure 1:



21

1 It shows a number of CPUs (processor cores), each with its own memory. The Host uses Data
2 Flow Control Unit 115 to communicate with the memories. Data Flow Control Unit 115 is more
3 than just a multiplexor, it allows the host to write to all the memories simultaneously.
4
5 From Column 2, lines 44 – 50:
6 Another aspect of the disclosed method and apparatus is a data flow control unit that
7 controls the flow of input and output data to each of the embedded memory arrays. This
8 device broadcasts the test program to each of the embedded memory arrays at the same time
9 thereby enabling the simultaneous testing of multiple embedded memory arrays.
10
11 As noted, each processor has its own memory. CPU 1 (105) can access only Embedded Memory
12 110. CPU 2 (106) can access only Embedded Memory 111. CPU 3 (107) can access only
13 Embedded Memory 112. Additional CPUs such as CPU n (108) can access only the Embedded
14 Memory (113) that is associated with it.
15
16 Although Jarboe consistently refers to the Embedded Memory only as *memory* it is apparent that
17 the memory is a RAM.
18 1. The Background of the invention (Column 1, lines 36 – 42) refers to SRAMs:
19 As is known in the art, a memory cache often accompanies each processor on a chip. This
20 memory cache enables the processor to operate at maximum efficiency by reducing the
21 time required to retrieve data from memory locations outside of the chip. The memory
22 cache associated with a processor is commonly an array of Static Random Access
23 Memory ("SRAM") devices.
24
25 2. If the Embedded Memory were ROM the Data Flow Control Unit would not be writing
26 to it and the only test needed would be to verify its contents.
27
28 As noted above, from Column 2, lines 44 – 50, the test program for each Embedded Memory is
29 loaded into the Embedded Memory.
30
31 There is a problem using a processor to test its own memory when it has only that one memory.
32
33 While it is easy to detect errors caused by memory array nodes that are permanently stuck
34 (“hard” errors) RAMs are also subject to errors caused by pattern sensitivity, which means that

1 the data in one cell can be altered by the pattern of data in other cells ("soft" errors). See
2 Appendix C: **Semiconductor Memories - A Handbook of Design, Manufacture, and**
3 **Application**, Second Edition, by Betty Prince, © 1983, 1991 by John Wiley & Sons, Ltd., pages
4 700 – 708. Note that Ms. Prince is associated with Texas Instruments which is also the Assignee
5 of the Jarboe patent.
6
7 When the memory being tested also contains the test program it is impossible to load the selected
8 test pattern into the entire memory. Otherwise, it will overwrite the test program and the program
9 will crash. As a result, Jarboe's method cannot completely test the embedded memories for
10 pattern sensitivity.
11
12 Moving the test program to another part of memory while its original location is being tested
13 does not solve the problem. The result is still that the selected memory pattern cannot be written
14 into the entire memory.
15
16 This also applies to memory locations used for storing variables as well as the program stack.
17 This last problem can be handled if the processor has a sufficient number of registers to allow
18 them to be used for storing variables and if the code is written to not require a program stack (i.e.
19 no subroutines).
20
21 An example of where this was done is the *Hard Drivin'* coin-operated game by Atari Games
22 Corporation, produced in 1988. (See Appendix D)
23
24 The Main Processor is a Motorola 68010 with separate Program ROM and Program RAM. After
25 a Power-On Reset (or a Hard Reset) the 68010 starts executing the program out of Program
26 ROM. The first thing it does is look at the Self-Test Switch. If the Self-Test switch is not closed
27 the 68010 starts the game. If the switch is closed the 68010 first calculates the Program ROM
28 checksums and warns the test technician if they are incorrect. The next step is to perform the
29 Program RAM memory test using a variety of memory test patterns. This is done without the
30 program using the memory it is testing for its own housekeeping. The 68010 contains 16
31 registers. The memory test program uses some of them for storing variables. In order to not use
32 all in-line code for the test program, some of the registers are used as Return Addresses for
33 pseudo-subroutines. When it is desired to call a pseudo-subroutine the return address is written to

a register. Then the program jumps to the beginning of the pseudo-subroutine. After the pseudo-subroutine is done it jumps to the address in the register containing the return address. The use of software macros makes it unnecessary to explicitly code every instruction.

Even so, the use of pseudo-subroutines is not transparent to the User as is the use of a Program Stack, but it allows the Program RAM to be fully tested. After the Program RAM is verified as Good, then it is available to the Program for use in storing variables and as the Program Stack.

It should be noted that the current Applicant, Margolin, was the Project Engineer and Hardware Engineer for the Hard Drivin' project. He also specified and wrote most of the Self-Test code. This makes Margolin a Person Having Greater-than-Ordinary Skill in this Art.

3. U.S. Patent 6,035,380 **Integrated Circuit** issued March 7, 2000 to Shelton et al. ("Shelton").

Shelton teaches a single chip processor for use in a smart card. Smart cards are typically used to store information representing financial transactions involving money so security is extremely important. The programs are typically stored on ROM and use sophisticated encryption methods to prevent unauthorized persons from illegally adding money to the card.

Shelton adds an additional non-volatile memory to store programs (such as card games) that can be added from third-parties.

Shelton provides an architecture that prevents this third-party software from modifying the financial software or from modifying the financial data. This is done by segmenting the memory space by using memory paging and prohibiting third-party software from changing the memory pages; the instructions that change the memory page (privileged instructions) can only be executed from ROM, not from the pages in the non-volatile memory used for third-party software. Different third-party programs are on different pages in the non-volatile memory and are therefore prevented from interfering with each other.

Data is kept in a separate non-volatile memory and is also paged so that financial data cannot be modified by third-party software.

Shelton also adds a coprocessor (cryptographic logic unit 1101) used to perform encryption tasks in order to offload it from the main (slower) processor.

This is Shelton's description of his invention. From the Abstract:

> A single chip processor for use in a smart card has a plurality of instruction memory areas and a processor. Different instructions sets are selectively executable in response to a signal defining a memory area from which instructions are supplied. Preferably instruction and data memory areas are addressable as pages, wherein a page address cannot be directly accessed by a subset of instructions. The processor may include a central processing unit and a cryptographic logic unit which operate at different times and share common instruction memory and sequencing logic. Instructions are supplied to said cryptographic logic unit at an integer multiple of the rate at which they are supplied to said central processing unit.

Memory Architecture

From Column 8, lines 13 – 17:

> An improved memory arrangement for a smart card chip is shown in FIG. 7a. Instructions may be supplied from a privileged instruction memory 702, which is a read only memory area, or from a non-volatile instruction memory 703, which is an electrically erasable read only memory.

But only instructions coming from the ROM are allowed to change the memory page.

From Column 8, lines 30 – 46 (emphasis added):

> The privileged instruction memory 702 is a read only memory area, whose contents are defined before the card is manufactured and distributed. The non-volatile instruction memory 703 may have its contents changed after the card has been distributed. For example, instructions may be updated during an interactive session with the terminal 101 shown in FIG. 1, with new instructions supplied from the large computer 107 at a remote site. The central processing unit 701, shown in FIG. 7a, includes a selective instruction decoder 710. The central processing unit has a number of possible instructions, which are referred to as an instruction set. <u>The selective instruction decoder 710 only allows the full set of instructions to be executed from the privileged instruction memory 702</u>. Certain instructions, particularly those which modify or read any of the page registers 715, 717, 719 or 721, are prevented from being used when they are supplied from the non-volatile instruction memory 703.

Note that new software for the financial program may be added to the non-volatile memory but will not be able to execute privileged instructions.

From Column 8, line 59 – column 9 line 13:

> A representation of instructions stored in the two instruction memory areas 702 and 703 shown in FIG. 7a, is shown in FIG. 8. Operating system instructions 801 and serial communications instructions 802 are stored in read only memory in the privileged instruction memory 702. Only these instructions have full access to the instruction set, and hence the ability to change the contents of any of the page registers 715, 717, 719 and 721. Third party applications 803 and 804, which may have been received through a transfer at the terminal 101, are unable to access the full instruction set, because they are stored in the non-volatile memory 703.
>
> By preventing a third party application from changing a page register, or jump to an instruction in another memory area, privileged instructions may control the type of operations performed by third party applications. For example: if third party application 803 is stored in a portion or page of non-volatile instruction memory 703 indexed by a particular page register value, it cannot directly read or jump to an instruction in third party application 804, which is stored in a different page of non-volatile instruction memory 703. Furthermore, page registers 719 and 721 shown in FIG. 7a cannot be directly modified by a third party application.

Memory paging is done using the standard method used for paging memory. From Column 8, lines 18 – 22 (Referring to Figure 7a):

> A page register 715 defines the most significant eight bits of the address supplied to the privileged instruction memory 702, and an offset register defines the least significant eight bits of the address supplied to the privileged instruction memory 702.

The privileged instruction memory 702 (ROM) is organized as 512 x 128 bits. Multiplexors 1202, 1203, and 1204 allow the 128-bit very-long words to be selected as 32-bit long words, 16-bit words, or 8-bit bytes. This is shown in Figure 12 which is reproduced below.

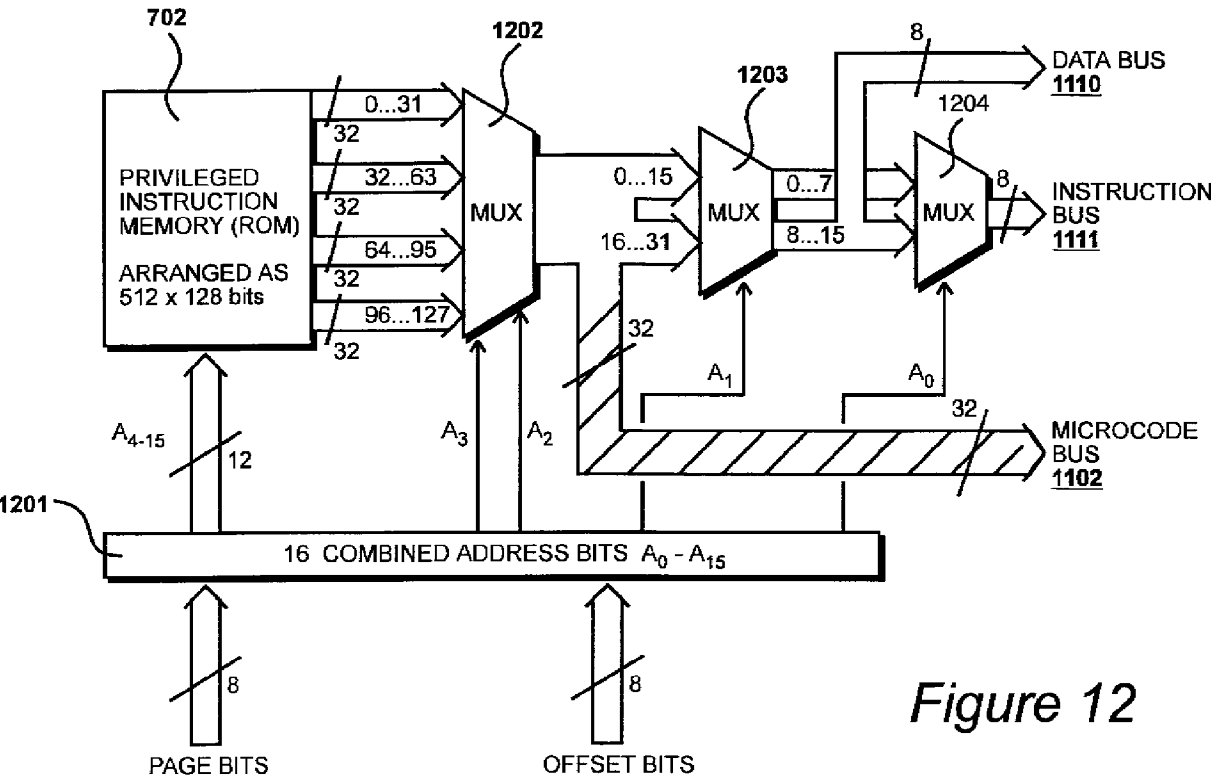


Figure 12

From Column 12, lines 52 – 65:

> The data from each location in the privileged instruction memory is considered as comprising four lots of thirty-two bit data words, thus forming a type of cache, which are fed to a four way multiplexer 1202. One of these four words is then selected as the output of the multiplexer 1202, according to the two address lines A3 and A2 which are supplied to it. In this way, a thirty-two bit data word may be selected with an access time of fifty nanoseconds, provided the thirty two bit word was already available within the one hundred and twenty-eight bit memory array output. If this is not the case, memory control logic

1 circuits automatically insert a wait state into the instruction fetch cycle, such that a different
2 one hundred and twenty-eight bit word may be fetched, which contains the desired thirty-
3 two bit word.
4
5 As shown in Figure 12, address bus 1201 is 16 bits (A0 – A15). The most significant 12 bits (A4
6 – A15) are used to address the ROM. This allows for 4096 very-long-words of 128 bits.
7 (Because the ROM contains only 512 very-long-words there is room for growth.)
8
9 MUX 1202 uses address bits A2 and A3 to select among four 32-bit long words out of the 128
10 data bits coming from ROM 702. (32 * 4 = 128).
11
12 After MUX 1202 the 32-bit long words can be used in different ways.
13
14 1). The 32-bit long words from MUX 1202 are divided into bytes and all bytes are sent to
15 Instruction Bus 1111.
16
17 MUX 1203 uses address bit A1 to further select between two 16-bit words from the 32-bit
18 long word selected by MUX 1202.
19
20 All 16 bits from MUX 1203 go to MUX 1204 which uses address A0 to select between
21 bytes containing bits (0..7) or (8..15). The byte selected by MUX 1204 goes to Instruction
22 Bus 1111.
23
24 2). The 32-bit long words from MUX 1202 are divided into bytes. All bytes are sent to
25 Instruction Bus 1111 but, in addition, the high byte (bits 8..15) is also sent to Data Bus 1110.
26
27 MUX 1203 uses address bit A1 to further select between two 16-bit words from the 32-bit
28 long word selected by MUX 1202.
29
30 All 16 bits from MUX 1203 go to MUX 1204 which uses address A0 to select between
31 bytes containing bits (0..7) or (8..15). The byte selected by MUX 1204 goes to Instruction
32 Bus 1111.
33
34 In addition, the high byte (bits 8..15) is also sent to Data Bus 1110.
35

3). The 32-bit long words from MUX 1202 are sent to Microcode Bus 1102 which goes to CLU 1101 (the cryptographic logic unit).

Communications

Shelton's smart card communicates with the outside world through communications terminals 302, which is a connector with contact pads:

From Column 5, lines 39 – 54:

> The card 201 includes communication terminals 302 allowing communication with external devices. In particular, these terminals include a terminal for receiving a two point seven to five point five volt power supply, a ground connection, a clock and a reset connection. These communication terminals 302 consists of flat, gold-plated areas of metal, which are fabricated in accordance with an international standard for smart cards. Thus cards may be interchangeable and facilitate data transfer in accordance with established protocols.
>
> The communication terminals 302 are electrically and bonded on the reverse side single silicon chip which is embedded within the smart card substrate. Only the communication terminals 302 are actually visible on the surface of the smart card, with the rest of the surface typically used for the cardholder's identity, and a company logo.

And from Column 7, lines 15 – 18:

> The operating system also communicates with instructions for serial communications 504, which provide the ability to transfer information to and from the outside world via the smart card terminals 302.

See Figure 3 below:

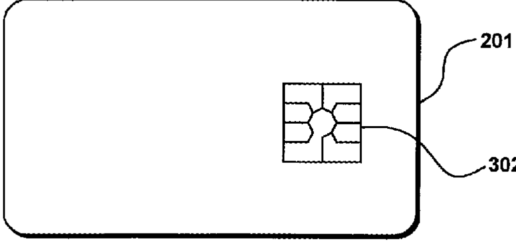


Figure 3

Figure 7A shows that this communications with the outside world goes through central processing unit 701.

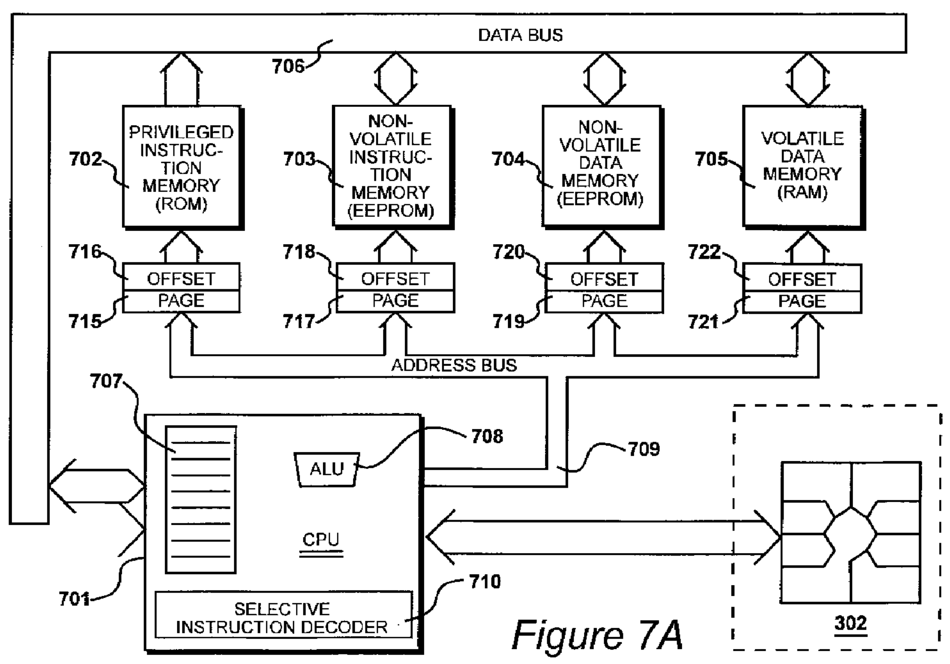


Figure 7A

The question may arise as to who (or what) Shelton considers a **User** and what he considers a ***User's System***.

Shelton's User is the person who uses the smart card and the User's System is the equipment that it is connected to through communications terminals 302 (the communications connector). Shelton shows two types of equipment that the smart card may be connected to.

1). The smart card may be connected to a smart card terminal which is attached to a mainframe computer. From Column 3, lines 61 – 62:

> Figure 1 shows a smart card terminal, a communications link, and a mainframe computer.

(In this case the communications link is between the smart card terminal and the mainframe computer.)

Figure 1 is reproduced below.

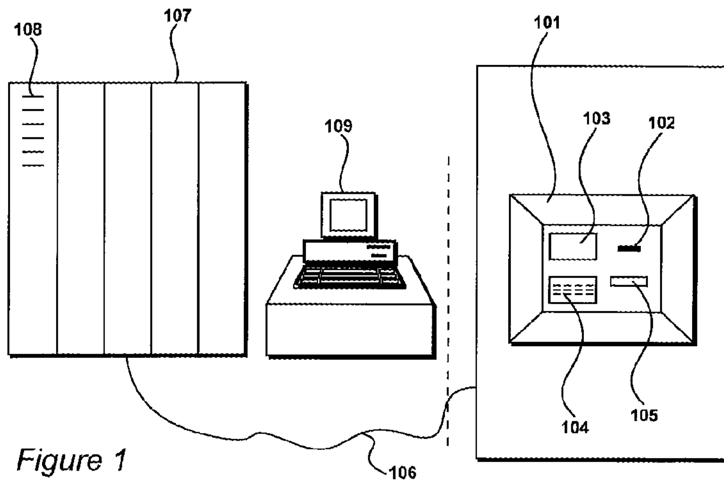


Figure 1

1
2
3 From Column 4, line 51 – Column 5, line 13 (emphasis added):
4
5 A terminal for allowing smart card transactions is shown in FIG. 1. The terminal 101
6 includes a slot 102 for receiving a smart card, a visual display unit 103 for providing the
7 ***smart card user*** with options and instructions for use. The ***user of the smart card*** responds
8 to displayed options and instructions by pressing buttons 104, which are arranged to enable
9 the user to provide numerical and functional input data. A cash slot 105 is provided in the
10 event that ***the user*** should wish to translate money represented by data stored on the smart
11 card into conventional cash.
12
13 The smart card terminal 101 communicates with a large computer 107 via a communications
14 link 106. The computer 107 includes a large amount of data storage capacity in the form of
15 arrays of hard disk drives 108. A computer terminal 109 enables an operator of the computer
16 107 to control access provided to ***smart card users*** via the smart card terminal 101. For
17 example, if a smart card is stolen, a computer operator may instruct the computer 107 not to
18 authorise any subsequent transfer of money to the card from the owner's account.
19
20 Column 5, lines 4 – 13 (emphasis added):
21
22 Smart cards may be used to exchange money tokens using appropriate equipment, such as
23 that shown in FIG. 1. Alternatively, a smaller terminal may be located alongside a
24 supermarket checkout counter, so that a smart card may be used instead of cash. Thus, when
25 paying for goods, the amount of money stored on the smart card is reduced. The terminal
26 shown in FIG. 1 may be used to transfer money from the ***user's account***, into the smart card.
27 In this way, the same smart card may be discharged and recharged with amounts of cash, at
28 the ***user's convenience***.
29
30 2). The smart card may also be used with equipment that allows funds to be transferred between
31 cards.

1

2 From Column 5, lines 14 – 26 (emphasis added):

3 Money may be exchanged directly from one smart card to another using the portable
4 exchange device shown in FIG. 2. The portable hand held exchange device is arranged to
5 receive a first smart card 201 and a second smart card 202. The device includes a keyboard
6 203 and a display device 204, providing a ***user interface*** to allow ***smart card users*** to insert
7 their cards into the device and to specify an amount of financial token data to be exchanged
8 between the smart cards, along with an indication of the direction of exchange. In addition,
9 the device may also be used by the respective parties to the transaction for them to enter
10 their personal identification numbers, as may be required in order to authorise a transaction
12 between cards 201 and 202.

14

16 Figure 2 is reproduced to the right.

18

20

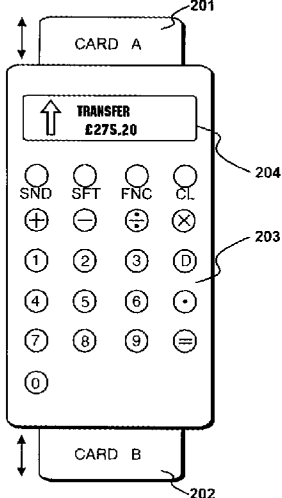


Figure 2

1 **Section 2 - Examiner's Rejections**

2

3 ***Claim Rejections - 35 USC § 112***

4 **Rejection 1.** Claims 1-16 were rejected under 35 U.S.C. 112, second paragraph, as being
5 indefinite for failing to particularly point out and distinctly claim the subject matter which
6 applicant regards as the invention.

7

8 Specifically, as per claims 1, 6 and 11, lines 2 - 4, the Examiner requested interconnection and/or
9 interrelation between the following elements and an explanation of what they are for:

10 a) a memory array;

11 b) a processor; and

12 c) a processor RAM memory.

13

14 As per claim 14, lines 2-4, the Examiner requested interconnection and/or interrelation between
15 the following steps and an explanation of what they are for:

16 a) providing a memory array;

17 b) providing a processor; and

18 c) providing a processor RAM memory.

19

20 Applicant has amended claims 1, 6, 11, and 14 as requested. Dependent claims 4, 7, 9, 12, and 13
21 have been canceled. Dependent Claims 2, 3, 5, 8, 10, 15, and 16 have been amended and should
22 be allowable as amended.

23

24 ***Claim Rejections - 35 USC § 102***

25

26 **Rejection 3.** Claims 1-2, 6-7, 11-12 and 14-15 were rejected under 35 U.S.C. 102(b) as being
27 anticipated by Shelton et al (6,035,380).

28 *As per claim 1, Shelton et al disclose a single chip integrated circuit, comprising:*

29 *(a) a memory array (102);*

30 *(b) a processor (701);*

31 *(c) a processor RAM memory (705);*

(d) a multiplexor (1202);

whereas said multiplexor controls and arbitrates access between said memory array, said processor, said processor RAM memory, and a user's system. (See Figs. 11 and 12, col. 11, lines 22-45 and col. 12, lines 10-65).

Applicant's Response:

1. Shelton's element 102 is not a memory array. It is a slot for receiving a smart card.

From Column 4, lines 51 – 54 (emphasis added):

> A terminal for allowing smart card transactions is shown in FIG. 1. The terminal 101 includes a slot 102 for receiving a smart card, a visual display unit 103 for providing the smart card user with options and instructions for use. The user of the smart card responds to displayed options and instructions by pressing buttons 104, which are arranged to enable the user to provide numerical and functional input data. A cash slot 105 is provided in the event that the user should wish to translate money represented by data stored on the smart card into conventional cash.

Presumably, the Examiner meant to refer to Privileged Instruction Memory (ROM) 702. If this is not the case, Applicant requests clarification.

2. Shelton's Privileged Instruction Memory (ROM) 702 is not a RAM (Read-Write Memory) as is Applicant's memory array.

From Applicant's specification:

> **[0031]** Referring to Figure 1, Memory Array 106 may take several forms. It may be a conventional read/write memory comprising row and address decoders, a memory cell array, and sense amplifiers. The memory cell array may be dynamic or static.

3. Shelton's Multiplexor 1202 is part of a chain of multiplexors (Multiplexors 1202, 1203, and 1204) which allow the 128-bit very-long words produced by Privileged Instruction Memory (ROM) 702 to be selected as 32-bit long words, 16-bit words, or 8-bit bytes. This is shown in Figure 12.

The privileged instruction memory 702 (ROM) is organized as 512 x 128 bits.

1 From Column 12, lines 52 – 65:
2
3 The data from each location in the privileged instruction memory is considered as
4 comprising four lots of thirty-two bit data words, thus forming a type of cache, which are
5 fed to a four way multiplexer 1202. One of these four words is then selected as the output of
6 the multiplexer 1202, according to the two address lines A3 and A2 which are supplied to it.
7 In this way, a thirty-two bit data word may be selected with an access time of fifty
8 nanoseconds, provided the thirty two bit word was already available within the one hundred
9 and twenty-eight bit memory array output. If this is not the case, memory control logic
10 circuits automatically insert a wait state into the instruction fetch cycle, such that a different
11 one hundred and twenty-eight bit word may be fetched, which contains the desired thirty-
12 two bit word.
13
14 As shown in Figure 12, address bus 1201 is 16 bits (A0 – A15). The most significant 12 bits (A4
15 – A15) are used to address the ROM. This allows for 4096 very-long-words of 128 bits.
16 (Because the ROM contains only 512 very-long-words there is room for growth.)
17
18 MUX 1202 uses address bits A2 and A3 to select among four 32-bit long words out of the 128
19 data bits coming from ROM 702. (32 * 4 = 128).
20
21 After MUX 1202 the 32-bit long words can be used in different ways.
22
23 1). The 32-bit long words from MUX 1202 are divided into bytes and all bytes are sent to
24 Instruction Bus 1111.
25
26 MUX 1203 uses address bit A1 to further select between two 16-bit words from the 32-bit
27 long word selected by MUX 1202.
28
29 All 16 bits from MUX 1203 go to MUX 1204 which uses address A0 to select between
30 bytes containing bits (0..7) or (8..15). The byte selected by MUX 1204 goes to Instruction
31 Bus 1111.
32
33 2). The 32-bit long words from MUX 1202 are divided into bytes. All bytes are sent to
34 Instruction Bus 1111 but, in addition, the high byte (bits 8..15) is also sent to Data Bus 1110.
35

1 MUX 1203 uses address bit A1 to further select between two 16-bit words from the 32-bit
2 long word selected by MUX 1202.
3
4 All 16 bits from MUX 1203 go to MUX 1204 which uses address A0 to select between
5 bytes containing bits (0..7) or (8..15). The byte selected by MUX 1204 goes to Instruction
6 Bus 1111.
7
8 In addition, the high byte (bits 8..15) is also sent to Data Bus 1110.
9
10 3). The 32-bit long words from MUX 1202 are sent to Microcode Bus 1102 which goes to CLU
11 1101 (the cryptographic logic unit).
12
13 As has been explained, Shelton's Multiplexor 1202 is used only to select among four 32-bit long
14 words out of the 128 data bits coming from Privileged Instruction Memory (ROM) 702. Its
15 outputs go only to Multiplexor 1203 and Microcode Bus 1102.
16
17 Multiplexor 1202 does not control and arbitrate access between Privileged Instruction Memory
18 (ROM) 702, CPU 701, Volatile Data Memory 705, and Shelton's User's System.
19
20 Indeed, even the chain of Multiplexors 1202, 1203, and 1202 do not control and arbitrate access
21 between Privileged Instruction Memory (ROM) 702, CPU 701, Volatile Data Memory 705, and
22 Shelton's User's System.
23
24 Neither Figure 11 nor Figure 12 show Shelton's User's System.
25
26 Shelton's User is the person who uses the smart card and the User's System is the equipment that
27 it is connected to through communications terminals 302 (the communications connector).
28
29 Shelton shows two types of equipment that the smart card may be connected to.
30

1). The smart card may be connected to a smart card terminal which is attached to a mainframe computer. From Column 3, lines 61 – 62:

> Figure 1 shows a smart card terminal, a communications link, and a mainframe computer.

(In this case the communications link is between the smart card terminal and the mainframe computer.)

See Figure 1.

From Column 4, line 51 – Column 5, line 13 (emphasis added):

> A terminal for allowing smart card transactions is shown in FIG. 1. The terminal 101 includes a slot 102 for receiving a smart card, a visual display unit 103 for providing the ***smart card user*** with options and instructions for use. The ***user of the smart card*** responds to displayed options and instructions by pressing buttons 104, which are arranged to enable the user to provide numerical and functional input data. A cash slot 105 is provided in the event that ***the user*** should wish to translate money represented by data stored on the smart card into conventional cash.
>
> The smart card terminal 101 communicates with a large computer 107 via a communications link 106. The computer 107 includes a large amount of data storage capacity in the form of arrays of hard disk drives 108. A computer terminal 109 enables an operator of the computer 107 to control access provided to ***smart card users*** via the smart card terminal 101. For example, if a smart card is stolen, a computer operator may instruct the computer 107 not to authorise any subsequent transfer of money to the card from the owner's account.

Column 5, lines 4 – 13 (emphasis added):

> Smart cards may be used to exchange money tokens using appropriate equipment, such as that shown in FIG. 1. Alternatively, a smaller terminal may be located alongside a supermarket checkout counter, so that a smart card may be used instead of cash. Thus, when paying for goods, the amount of money stored on the smart card is reduced. The terminal shown in FIG. 1 may be used to transfer money from the ***user's account***, into the smart card. In this way, the same smart card may be discharged and recharged with amounts of cash, at the ***user's convenience***.

2). The smart card may also be used with equipment that allows funds to be transferred between cards.

From Column 5, lines 14 – 26 (emphasis added):

> Money may be exchanged directly from one smart card to another using the portable exchange device shown in FIG. 2. The portable hand held exchange device is arranged to receive a first smart card 201 and a second smart card 202. The device includes a keyboard

1 203 and a display device 204, providing a ***user interface*** to allow ***smart card users*** to insert
2 their cards into the device and to specify an amount of financial token data to be exchanged
3 between the smart cards, along with an indication of the direction of exchange. In addition,
4 the device may also be used by the respective parties to the transaction for them to enter
5 their personal identification numbers, as may be required in order to authorise a transaction
6 between cards 201 and 202.
7
8 See Figure 2.
9
10 Shelton's smart card communicates with the outside world through communications terminals
11 302, which is a connector with contact pads:
12
13 From Column 5, lines 39 – 54:
14 The card 201 includes communication terminals 302 allowing communication with external
15 devices. In particular, these terminals include a terminal for receiving a two point seven to
16 five point five volt power supply, a ground connection, a clock and a reset connection.
17 These communication terminals 302 consists of flat, gold-plated areas of metal, which are
18 fabricated in accordance with an international standard for smart cards. Thus cards may be
19 interchangeable and facilitate data transfer in accordance with established protocols.
20
21 The communication terminals 302 are electrically and bonded on the reverse side single
22 silicon chip which is embedded within the smart card substrate. Only the communication
23 terminals 302 are actually visible on the surface of the smart card, with the rest of the
24 surface typically used for the cardholder's identity, and a company logo.
25
26 And from Column 7, lines 15 – 18:
27
28 The operating system also communicates with instructions for serial communications 504,
29 which provide the ability to transfer information to and from the outside world via the smart
30 card terminals 302.
31
32 See Figure 3.
33
34 Figure 7A shows that this communications with the outside world goes through central
35 processing unit 701.
36
37 Thus, Multiplexor 1202 does not control and arbitrate access between Privileged Instruction
38 Memory (ROM) 702, CPU 701, Volatile Data Memory 705, and Shelton's User's System.
39
40 In addition, Multiplexor 1202 is only a data selector. It does not arbitrate access between
41 anything.

1 From Applicant's paragraph 0038:

2 **[0038]** Multiplexor MUX 101 controls and arbitrates access between the internal
3 programmable processor, Memory Array 106, and the User's system. It allows TCPU 103 to
4 access Memory Array 106. It also allows Memory Array 106 to be accessed by external
5 buses such as when the invention is used as main memory in a User's system. In addition,
6 through the use of the RS input on MUX 101, MUX 101 allows the User's system to control
7 TCPU 103, access TCPU RAM Memory 104 in order to load the program to be run by
8 TCPU 103, and access Non-Volatile Memory 105. The BUSY output on MUX 101 tells the
9 User's system that Memory Array 106 is being used by TCPU 103 and to wait. The TCPU
10 Interface contains a similar signal to tell TCPU 103 that Memory Array 106 is being used by
11 the User's system and to wait. The preferred memory arbitration scheme is to give the
12 User's system priority to Memory Array 106. If TCPU 103 is accessing Memory Array 106
13 at the beginning of a User system access, the User system waits until the next memory cycle
14 at which point TCPU 103 is stalled and the User system gets access to Memory Array 106.
15 The CLOCK input to MUX 101 is used by TCPU 103 when the invention is used by a
16 User's system in order to avoid the potential for conflicts caused by metastable instability of
17 an arbitration logic circuit that would exist if TCPU 103 used a clock having a frequency not
18 synchronized to the clock used by the User's system. During Wafer testing, the CLOCK
19 input to MUX 101 may be used as a reference by Programmable Clock 102. In Figure 1,
20 MUX 101 provides external access to Memory Array 106 through a non-multiplexed
21 address bus.
22
23 Shelton's Multiplexor 1202 provides no such limitation. Applicant's amended claim 1 traverses
24 Shelton.
25
26 The Examiner has rejected dependent claim 2, *Shelton et al further comprising a non-volatile*
27 *memory (704).* Applicant has traversed amended independent claim 1 over Shelton making claim
28 2 allowable as amended.
29
30 The Examiner has rejected independent claims 6, 11 and 14, *these claims are rejected under*
31 *similar rationale as set forth in claim 1.* Applicant traverses amended independent claims 6, 11,
32 and 14 over Shelton under the same rationale as set forth in traversing Shelton over amended
33 claim 1.
34
35 The Examiner has rejected dependent claims 7, 12 and 15, *these claims are rejected under*
36 *similar rationale as set forth in claim 2.* Applicant has canceled dependent claims 7 and 12.
37 Applicant has traversed the amended independent claim 14 over Shelton making dependent claim
38 15 allowable as amended.

1 **Rejection 5.** Claims 3, 8, 13 and 16 were rejected under 35 U.S.C. 103(a) as being unpatentable
2 over Shelton et al (6,035,380) in view of Fulks et al (4,194,113).

3 *As per claim 3, the teaching of Shelton et al have been discussed above.*
4 *Shelton et al further disclose a clock circuit (Fig. 14). They do not disclose that the*
5 *clock circuit is a programmable clock. However, Fulks et al disclose a programmable clock*
6 *(153). (See Fig. 5, col. 14, lines 53-68). Therefore, it would have been obvious to a person*
7 *of ordinary skill in the art, at the time the invention was made, to incorporate the*
8 *programmable clock as taught by Fulks et al into the clock circuit of Shelton et al so that it*
9 *can produce a high speed programmable clock signals utilized to control the operation of a*
10 *high speed processor (HSP). (See col. 14, lines 53-68).*
11
12 *As per claims 8, 13 and 16, these claims are rejected under similar rationale as set*
13 *forth in claim 3.*
14
15
16 Applicant's Response:
17
18 1. Shelton has been traversed above in Rejection 3 above.
19
20 2. The Fulks patent is for a computer-controlled digital signature analyzer used to isolate faults
21 on digital printed circuit boards. The computer stores the signatures and other information about
22 the board being tested and starts by performing a go/no-go test through the board's edge
23 connector. If the board fails, the computer directs the operator where to place a probe to read the
24 signatures of particular circuit nodes. It then compares the signature of the nodes to the signature
25 of the same nodes obtained from a known-good printed circuit board. See Column 6, lines 3 – 6,
26 Column 6, lines 29 – 38, Column 22, lines 40 – 49, and Figure 1A.
27
28 The central part of Fulks' invention uses signature analysis. One of the advantages of signature
29 analysis is that it does not require the storage of the entire data stream at every node, and which
30 therefore require very large memory storage capability. See Column 25 lines 28 – 31.
31
32 One of the shortcomings of signature analysis is that feedback loops must be broken. See
33 Column 25, lines 16 – 32. Fulks solves this problem by storing a smaller data stream. See
34 Column 25, lines 33 – 43.
35
36 Fulks uses a separate high-speed processor (HSP 29 in Figure 2) to interface to the printed circuit
37 board's connector. This is succinctly explained in application 05/895,898 (now U.S. Patent

1 4,196,386 **Method and portable apparatus for testing digital printed circuit boards** issued
2 April 1, 1980 to Phelps) incorporated by reference in Fulks. See Phelps Column 9, lines 44 – 58:

> 3 According to the present invention, a high speed processor (HSP), which is programmable to
> 4 operate at many times the speed of the comparatively slow main processor is utilized to
> 5 operate in response to and simultaneously with the main processor to sequentially route data
> 6 received in parallel format from the main processor to predetermined ones of the 192
> 7 driver/sensor circuits. The fast sequential outputting is accomplished by means of a
> 8 subroutine, referred to herein as an "H" file, in the form of object code for the HSP and
> 9 stored in the HSP memory. This subroutine defines groups of specific pins of the board
> 10 under test as "destination busses" and sequentially routes sixteen bit data words each
> 11 contained in a respective single instruction in the test program to such "destination busses".

12
13 In Fulks, we find out that the HSP has a programmable clock. In Column 14, lines 53 – 68:
14

> 15 Still referring to FIG. 5, clock circuit 153 includes a number of conventional counters,
> 16 registers, flip flops, and some control gating circuitry to produce high speed programmable
> 17 clock signals utilized to control the operation of the HSP. The range of the cycle times of the
> 18 programmable clock signals is from 150 nanoseconds to approximately 12.5 microseconds,
> 19 in 50 nanosecond increments. Clock circuit 153 includes a register which is loaded from
> 20 main bus 27 to determine the programmable cycle time of the HSP. The cycle time of the
> 21 HSP controls the rate at which PSP 10 switches from one input/output pin to another during
> 22 the testing of the board under test. Clock circuit 153 may be readily implemented utilizing
> 23 Texas Instruments 74LS175 latches, 74S74 flip flops, and 74S161 counters as the main
> 24 components thereof.

25
26 Thus, Fulks' programmable clock is used with the HSP 153 which contains drivers/sensors for
27 interfacing with the pins on the edge connector for a printed circuit board.
28
29 Fulks' invention could not be more different from Applicant's invention.
30
31 1). Fulks's invention is embodied in an instrument (and a fair-sized instrument at that; see
32 Figure 1A) for testing printed circuit boards using signature analysis and contains a limited
33 amount of storage for storing data streams.
34
35 2). Signature analysis compares the signatures from the board under test to the signatures
36 obtained from a known-good board.
37
38 3). Signature analysis is not used for testing RAMs.
39

4). Applicant's invention is a single chip memory with an embedded processor (TCPU 103) with its own separate memory (TCPU RAM 104) for testing the RAM (Memory Array 106), all in Applicant's Figure 1. The memory test program performed by TCPU 103 does not use signature analysis and does not store data streams; it performs an algorithmic test of Memory Array 106. See Table 1(a) through Table 1(k). The Applicant's invention is not a test instrument and does not test printed circuit boards.

Thus, although Fulks shows a programmable clock with HSP 153, Fulks teaches away from Applicant's invention.

Rejection 6. Claims 4-5, and 9-10 were rejected under 35 U.S.C. 103(a) as being unpatentable over Shelton et al (6,035,380) and further in view of Jarboe, Jr. et al (7,155,637).

> *As per claims 4-5, the teaching of Shelton et al have been discussed above. They do not disclose that a program is used by the processor to test the memory array. However, Jarboe, Jr. et al disclose that the program is used by the processor to test the memory array (col. 5, lines 53-63). Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the program that is used by the processor to test the memory array as taught by Jarboe, Jr. et al into the invention of Shelton et al so that test program can be used to test memory array to detect errors and the test program can be stored in the processor RAM memory for relater used.*

Applicant's Response:

1. Shelton has been traversed above in Rejection 3 above.

2. Jarboe's test program is used with memories containing redundant rows and columns that can be activated to replace defective rows and columns discovered during testing.

From the Abstract:

> The disclosed method and apparatus enables the testing of multiple embedded memory arrays associated with multiple processor cores on a single computer chip. According to one aspect, the disclosed method and apparatus identifies certain rows and columns within each of the embedded memory arrays that need to be disabled and also identifies certain redundant rows and columns in the embedded memory array to be activated. According to another aspect, the disclosed method and apparatus generates a map indicating where each of the memory failures occurs in each embedded memory array. If the testing process determines that the embedded memory array cannot be repaired, then a signal is provided

1 directly to an external testing device indicating that the embedded memory array is non-
2 repairable. Similarly, if the testing process determines that the failures in the embedded
3 memory array can be repaired, then a signal is provided directly to an external testing
4 apparatus indicating that the embedded memory array is repairable. Lastly, if no failures are
5 found in an embedded memory array, then a signal is provided to an external testing
6 apparatus indicating that the embedded memory array contains no failures.
7
8 There is no suggestion that Fulks' memory array contains redundant rows and columns that can
9 be used to repair defects in the memory array.
10
11 Therefore, there would be no motivation to use Jarboe's memory test program in Fulks.
12
13

1 **Section 3.**

2

3 For the foregoing reasons, Applicant submits that all objections and rejections have been
4 overcome. Applicant requests that the rejection of pending claims 1, 2, 3, 5, 6, 8, 10, 11, 14, 15,
5 and 16 be withdrawn and that the application be allowed as amended.

6

7 Respectfully submitted,

8

9 /Jed Margolin/ Date: September 20, 2007
10 Jed Margolin

11

12

13 Jed Margolin
14 1981 Empire Rd.
15 Reno, NV 89521-7430
16 (775) 847-7845

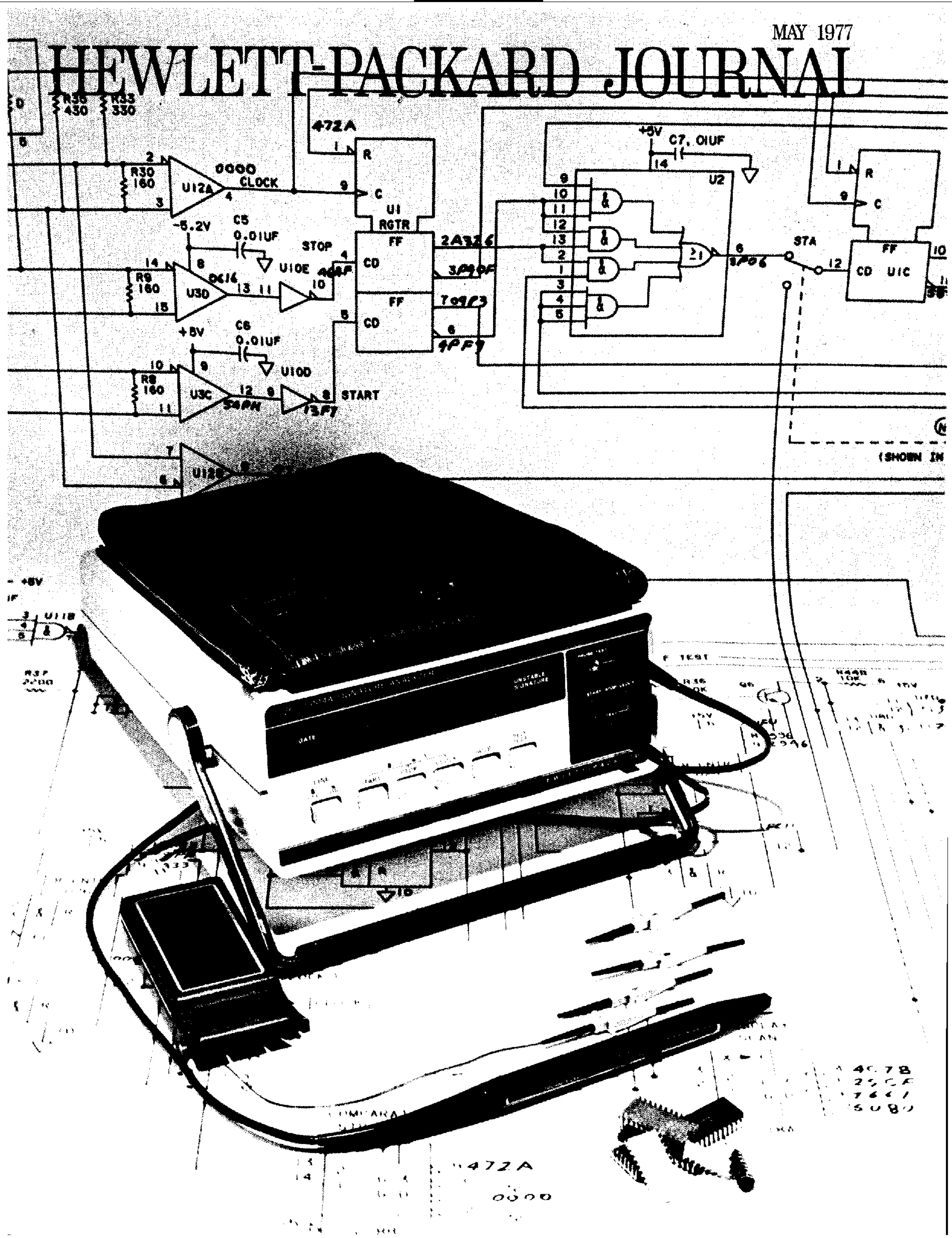
17

18 ___

19

20

Appendix A



Signature Analysis: A New Digital Field Service Method

In a digital instrument designed for troubleshooting by signature analysis, this method can find the components responsible for well over 99% of all failures, even intermittent ones, without removing circuit boards from the instrument.

by Robert A. Frohwerk

WITH THE ADVENT OF MICROPROCESSORS and highly complex LSI (large-scale integrated) circuits, the engineer troubleshooting digital systems finds himself dealing more with long digital data patterns than with waveforms. As packaging density increases and the use of more LSI circuits leaves fewer test points available, the data streams at the available test points can become very complex. The problem is how to apply some suitable stimulus to the circuit and analyze the resulting data patterns to locate the faulty component so that it can be replaced and the circuit board returned to service.

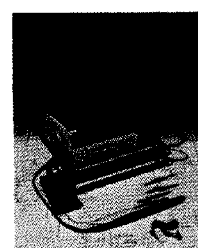
The search for an optimal troubleshooting algorithm to find failing components on digital circuit boards has taken many directions, but all of the approaches tried have had at least one shortcoming. Some simply do not test a realistic set of input conditions, while others perform well at detecting logical errors and stuck nodes but fail to detect timing-related problems. Test systems capable of detecting one-half to two-thirds of all possible errors occurring in a circuit have been considered quite good. These systems tend to be large, for factory-based use only, and computer-driven, requiring program support and software packets and hardware interfaces for each type of board to be tested. Field troubleshooting, beyond the logic-probe capability to detect stuck nodes, has been virtually neglected in favor of board exchange programs.

The problem seems to be that test systems have too often been an afterthought. The instrument designer leaves the test procedure to a production test engineer, who seeks a general-purpose solution because he lacks the time to handle each case individually.

Obviously it would be better if the instrument designer provided for field troubleshooting in his original design. Who knows a circuit better than its original designer? Who has the greatest insight as to how to test it? And what better time to modify a circuit to accommodate easy testing than before the circuit is in production?

New Tools Needed

But here another problem arises: what do we offer the circuit designer for tools? A truly portable test instrument, since field troubleshooting is our goal, would be a passive device that merely looked at a circuit and told us why it was failing. The tool would provide no stimulus, require little software support, and have accuracy at least as great as that of computer-driven factory-based test systems.



Cover: *Those strange-looking strings of four alphanumeric characters on the instrument's display and the schematic diagram are signatures, and the instrument is the 5004A Signature Analyzer, a troubleshooting tool for field repair of digital systems. With a failing system operating in a self-stimulating test mode, the service person probes various test points, looking for incorrect signature displays that can point to faulty components.*

In this Issue:

Signature Analysis: A New Digital Field Service Method, by Robert A. Frohwerk .... **page 2**
Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits, by Anthony Y. Chan .. **page 9**
Signature Analysis—Concepts, Examples, and Guidelines, by Hans J. Nadig .... **page 15**
Personal Calculator Algorithms I: Square Roots, by William E. Egbert .. **page 22**

 © Hewlett-Packard Company, 1977

If a tester provides no stimulus, then the circuit under test must be self-stimulating. Whereas this seemed either impossible or at best very expensive in the past, a self-stimulating circuit is not out of the question now. More and more designs are microprocessor-oriented or ROM-driven, so self-stimulus, in the form of read-only memory, is readily available and relatively inexpensive.

By forcing a limitation on software, we have eliminated the capability to stop on the first failure and must use a burst-mode test. Another restriction we will impose is that the device under test must be synchronous, in the sense that at the time the selected clock signal occurs the data is valid; not an unfair condition by any means, and it will be justified in the article beginning on page 15.

There are only a few known methods for compressing the data for a multiple-bit burst into a form that can be handled easily by a portable tester without an undue amount of software. One method used in large systems is transition counting. Another method, a much more efficient data compression technique borrowed from the telecommunications field, is the cyclic redundancy check (CRC) code, a sort of checksum, produced by a pseudorandom binary sequence (PRBS) generator.

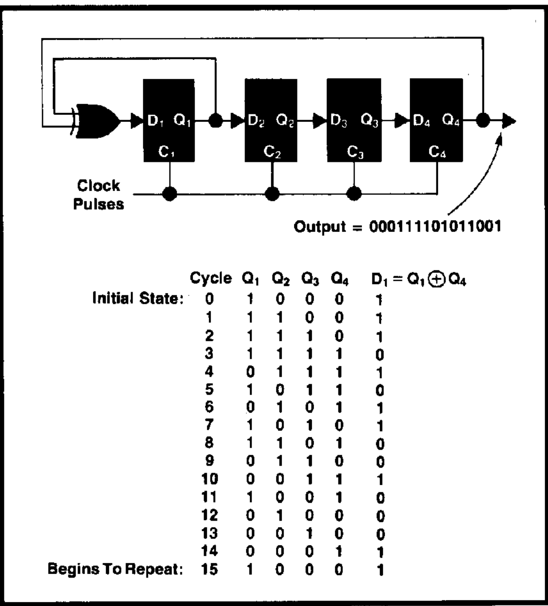
A troubleshooting method and a portable instrument based on this concept turns out to be the answer we are seeking. We call the method signature analysis and the instrument the 5004A Signature Analyzer. The instrument is described in the article on page 9. Here we will present the theory of the method and show that it works, and works very well.

Pseudorandom Binary Sequences

A pseudorandom binary sequence is, as implied, a pattern of binary ones and zeros that appears to be random. However, after some sequence length the pattern repeats. The random-like selection of bits provides nearly ideal statistical characteristics, yet the sequences are usable because of their predictability. A PRBS based upon an n-bit generator may have any length up to 2^n-1 bits before repeating. A generator that repeats after exactly 2^n-1 bits is termed maximal length. Such a generator will produce all possible n-bit sequences, excluding a string of n zeros. As an example, let us take the sequence: 000111101011001. This is a fifteen-bit pattern produced by a four-bit maximal-length generator ($15=2^4-1$). If we were to wrap this sequence around on itself, we would notice that all possible non-zero four-bit patterns occur once and only once, and then the sequence repeats.

To construct a PRBS generator we look to the realm of linear sequential circuits, which is where the simplest generators reside mathematically. Here there exist only two types of operating elements. The first is a modulo-2 adder, also known as an exclusive-OR gate. The other element is a simple D-type flip-flop, which being a memory element behaves merely as a time delay of one clock period. By connecting flip-flops in series we construct a shift-register as in Fig. 1, and by taking the outputs of various flip-flops, exclusive-ORing them, and feeding the result back to the register input, we make it a feedback shift register that will produce a pseudorandom sequence. With properly chosen feedback taps, the sequence will be maximal length. The fifteen-bit sequence above was produced by the generator in Fig. 1, with the flip-flops initially in the 0001 state since the all-zero state is disallowed. The table in Fig. 1 shows the sequence in detail. The list contains each of the sixteen ways of arranging four bits, except four zeros.

If we take the same feedback shift register and provide it with an external input, as in Fig. 2, we can overlay data onto the pseudorandom sequence. The overlaid data disturbs the internal sequence of the generator. If we begin with an initial state of all zeros and supply a data impulse of 1000..., the result is the same sequence as in Fig. 1 delayed by one clock period.



	Cycle	Q_1	Q_2	Q_3	Q_4	$D_1 = Q_1 \oplus Q_4$
Initial State:	0	1	0	0	0	1
	1	1	1	0	0	1
	2	1	1	1	0	1
	3	1	1	1	1	0
	4	0	1	1	1	1
	5	1	0	1	1	0
	6	0	1	0	1	1
	7	1	0	1	0	1
	8	1	1	0	1	0
	9	0	1	1	0	0
	10	0	0	1	1	1
	11	1	0	0	1	0
	12	0	1	0	0	0
	13	0	0	1	0	0
	14	0	0	0	1	1
Begins To Repeat:	15	1	0	0	0	1

Fig. 1. *Signature analysis is a troubleshooting technique that makes use of the cyclic redundancy check (CRC) code, a sort of checksum, produced by a pseudorandom binary sequence (PRBS) generator. Shown here is a feedback shift register that generates a 15-bit PRBS. The outputs of the four flip-flops go through all possible non-zero four-bit patterns and then the sequence repeats.*

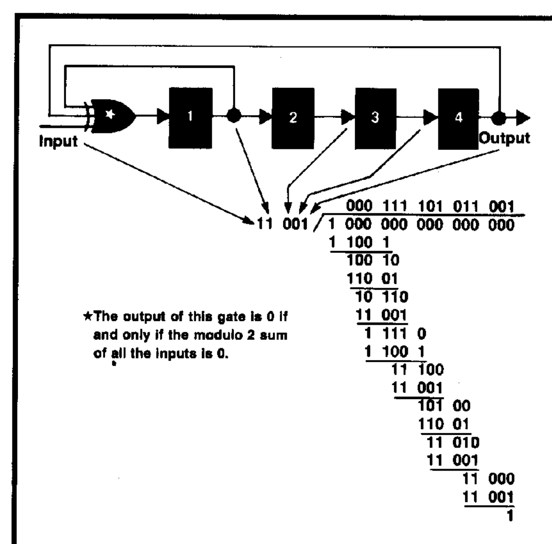


Fig. 2. *When the feedback shift register of Fig. 1 is provided with an external input, data can be overlaid on the PRBS generated by the circuit. Feeding data into a PRBS generator is the same as dividing the data by the characteristic polynomial of the generator.*

Shift Register Mathematics

A shift register may be described using a transform operator, D, defined such that $X(t) = DX(t-1)$. Multiplying by D is equivalent to delaying data by one unit of time. (Recall that we are concerned only about synchronous logic circuits.) In Fig. 2 the data entering the register is the sum of samples taken after one clock period and four clock periods along with the input data itself. Thus, the feedback equation may be written as $D^4X(t) + DX(t) + X(t)$ or simply X^4+X+1.

It happens that feeding a data stream into a PRBS generator is equivalent to dividing the data stream by the characteristic polynomial of the generator. For the particular implementation of the feedback shift register considered here the characteristic polynomial is X^4+X^3+1, which is the reverse of the feedback equation. Fig. 2 shows the register along with longhand division of the impulse data stream (100...). Keep in mind that in modulo-2 arithmetic, addition and subtraction are the same and there is no carry. It can be seen that the quotient is identical to the pattern in Fig. 1 and repeats after fifteen bits (the "1" in the remainder starts the sequence again).

Because the shift register with exclusive-OR feedback is a linear sequential circuit it gives the same weight to each input bit. A nonlinear circuit, on the other hand, would contain such combinatorial devices as AND gates, which are not modulo-2 operators and which would cancel some inputs based upon prior bits. In other words a linear polynomial is one for which $P(X+Y) = P(X) + P(Y)$. Take the example of Fig. 3, where the three different bit streams X, Y, and X+Y are fed to the same PRBS generator. Notice that the output sequences follow the above relationship, that is, $Q(X+Y) = Q(X) + Q(Y)$. Also, notice that Y is a single impulse bit delayed in time with respect to the other sequences and the only difference between X and X+Y is that single bit. Yet, Q(X+Y) looks nothing like Q(X). Indeed, if we stop after entering only twenty bits of the sequences and compare the remainders, or the residues in the shift register, they would be: $R(X+Y) = 0100$, $R(X) = 0111$.

Error Detection by PRBS Generator

Looking at this example in another manner, we can think of X as a valid input data stream and X + Y as an erroneous input with Y being the error sequence. We will prove later that any single-bit error, regardless of when it occurs, will always be detected by stopping the register at any time and comparing the remainder bits (four in this case) with what they should be. This error detection capability is independent of the length of the input sequence. In the example of Fig. 3, R(X+Y) differs from the correct R(X), and the effect of the error remains even though the error has disappeared many clock periods ago.

Let us stop for a moment to recall our original goal. We are searching for a simple data compression algorithm that would be efficient enough to be usable in a field service instrument tester. As such it was to require only minimal hardware and software support.

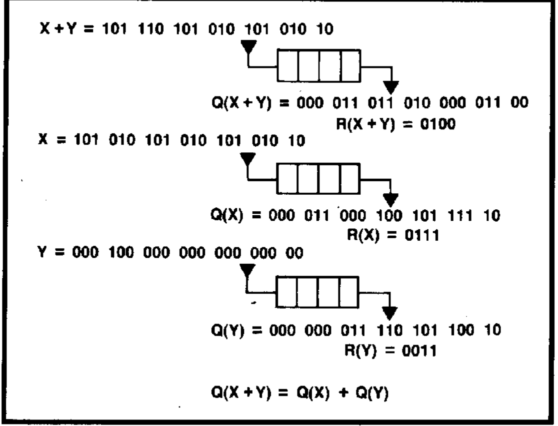


Fig. 3. *Three different input data sequences fed to the same PRBS generator produce very different output sequences even though the input sequences differ by only one bit. If the generators are stopped at some time and the patterns remaining in the flip-flops are compared, they are also different. These remainder patterns are called* signatures. *They show the effects of an error sequence Y added to a data stream X even when the error occurs only once in a long measurement window.*

We have now found such an algorithm. If the circuit designer arranges his synchronous circuit so as to provide clock and gate signals that produce a repeatable cycle for testing, then the feedback shift register is the passive device that we need to accumulate the data from a node in the instrument under test. By tracing through an instrument known to be good, the designer merely annotates his schematic, labeling each test point with the contents of the shift register at the end of the measurement cycle, and uses this information later to analyze a failing circuit. Because this PRBS residue depends on every bit that has entered the generator, it is an identifying characteristic of the data stream. We have chosen to call it a *signature*. The process of annotating schematics with good signatures as an aid in troubleshooting circuits that produce bad signatures has been termed *signature analysis*.

Errors Detected by Signature Analysis

We have claimed that any single-bit error will always be detected by a PRBS generator. But how about multiple errors? Also, our goal was to maintain error detection capability at least as good as existing methods. Earlier mention was made of transition counting, which appears to be the only other method that could easily be made portable. To show how signature analyis stands up against transition counting requires a mathematical discussion of the error detection capabilities of these methods. Take first the PRBS.

Assume X is a data stream of m bits, P is an n-bit PRBS generator, P^{-1} its inverse ($P^{-1}P = 1$), Q is a quotient and R the remainder.

$$P(X) = Q(X) \cdot 2^n + R(X). \qquad (1)$$

Take another m-bit sequence Y that is not the same as X and must therefore differ by another m-bit error sequence E such that

$$Y = X + E.$$

Now,

$$P(Y) = Q(Y) \cdot 2^n + R(Y)$$

so,

$$P(X+E) = Q(X+E) \cdot 2^n + R(X+E).$$

But all operators here are linear, so

$$P(X) + P(E) = Q(X) \cdot 2^n + Q(E) \cdot 2^n + R(X) + R(E).$$

Subtracting (or adding, modulo 2) with equation 1 above,

$$P(E) = Q(E) \cdot 2^n + R(E). \qquad (2)$$

However, if Y is to contain undetectable errors,

$$R(Y) = R(X).$$

It follows that

$$R(Y) = R(X+E) = R(X) + R(E) = R(X),$$
$$R(E) = 0.$$

Substituting into equation 2,

$$P(E) = Q(E) \cdot 2^n,$$

and all undetectable errors are found by

$$E = P^{-1}Q(E) \cdot 2^n. \qquad (3)$$

For a single-bit error

$$E = D^a(1)$$

where D is the delay operator, a is the period of the delay, and "1" is the impulse sequence 1000... Substituting into (3),

$$D^a(1) = P^{-1}Q(D^a(1)) \cdot 2^n.$$

D commutes with other linear operators, so

$$D^a(1) = D^a P^{-1}Q(1) \cdot 2^n$$
$$1 = P^{-1}Q(1) \cdot 2^n$$
$$P(1) = Q(1) \cdot 2^n.$$

But by the original assumptions,

$$P(1) = Q(1) \cdot 2^n + R(1)$$

and by addition

$$R(1) = 0.$$

However, it has been shown by example that $R(1) \neq 0$. Therefore, $E \neq D^a(1)$ and the set of undetectable errors E does not include single-bit errors; in other words, a single-bit error is always detectable. (An intuitive argument might conclude that a single-bit error would always be detected because there would never be another error bit to cancel the feedback.)

To examine all undetectable errors as defined by equation 3, it helps to consider a diagrammatical representation, Fig. 4, of:

$$E = P^{-1}Q(E) \cdot 2^n.$$

Since X, Y, and E are all m-bit sequences, it follows that $Q \cdot 2^n$ must be an m-bit sequence containing n final zeros. Q therefore contains (m−n) bits. Hence, there are 2^{m-n} sequences that map into the same residue as the correct sequence, and there are $2^{m-n}-1$ error sequences that are undetectable because they leave the same residue as the correct sequence. 2^m sequences can be generated using m bits and only one of these is correct, so the probability of failing to detect an error by a PRBS is

$$\text{Prob (PRBS, fail)} = \frac{\text{Undetectable Errors}}{\text{Total Errors}} = \frac{2^{m-n}-1}{2^m-1}.$$

For long sequences, large m,

$$\text{Prob (PRBS, fail)} \approx 1/2^n.$$

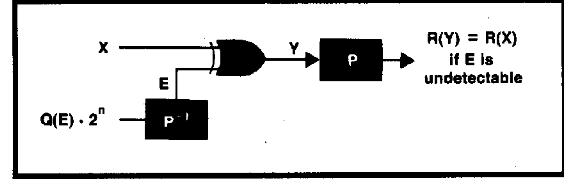


Fig. 4. *A diagrammatical representation of errors undetectable by signature analysis. For long data sequences the probability of not detecting an error approaches $1/2^n$, where n is the number of flip-flops in the feedback shift register.*

In summary, a feedback shift register of length n will detect all errors in data streams of n or fewer bits, because the entire sequence will remain in the register, $R(X) = P(X)$. For data streams of greater than n bits in length, the probability of detecting an error using a PRBS is very near certainty even for generators of modest length. The errors not detected are predictable and can be generated by taking all m-bit sequences with n trailing zeros and acting upon such sequences by the inverse of the n-bit PRBS generator polynomial P, that is

$$E = P^{-1}(Q \cdot 2^n).$$

Furthermore, such error detection methods will always detect a single-bit error regardless of the length of the data stream. It can also be proved that the only undetectable error sequence containing two errors such that the second cancels the effect of the first is produced by separating the two errors by exactly 2^n-1 zeros.[1] The one sequence of length n+1 that contains undetectable errors begins with an error and then contains other errors that cancel each time the original error is fed back.

Errors Detected by Transition Counting

It appears that signature analysis using a PRBS generator is a difficult act to follow, but let us give transition counting a chance. A transition counter assumes an initial state of zero and increments at each clock time for which the present data bit differs from the previous bit. With a transition counter the probability of an undetected error, given that there is some error, is:

$$\text{Prob (Trnsn, Fail)} = N_u/N_t,$$

where N_u = number of undetected errors and N_t = total number of errors. But

$$N_u = \sum_{r=0}^{m} p_{ur}$$

where p_{ur} = Prob (undetected errors given r transitions). However,

$$p_{ur} = N_{ur} \cdot p_r$$

where N_{ur} = number of undetected errors given r transitions, and p_r = Prob (counting r transitions). Reducing further,

$$N_{ur} = N_r - N_c,$$
$$p_r = N_r/N_s,$$

where N_c = number of ways of counting correctly (=1), N_s = total number of m-bit sequences, and N_r = number of ways of counting r transitions:

$$N_r = \binom{m}{r} = \frac{m!}{r!(m-r)!}$$

The binomial coefficient $\binom{m}{r}$ expresses the number of ways of selecting from m things r at a time. Looking back to the original denominator,

$$N_t = N_s - N_c.$$

Putting all of this together,

$$\text{Prob (Trnsn, Fail)} = \frac{\sum_{r=0}^{m} (N_r - N_c)(N_r/N_s)}{N_s - N_c}.$$

Or,

$$\text{Prob (Trnsn, Fail)} = \frac{\sum_{r=0}^{m} \left[\binom{m}{r} - 1\right] \binom{m}{r}/2^m}{2^m - 1}$$
$$\approx 1/\sqrt{m\pi}.$$

This is the probability of a transition counter's failing to detect an error in an m-bit sequence.

A similar argument finds the probability of the specific case where a single-bit error is not detected by a transition counter. There are 2^m sequences of m bits and any one of the m bits can be altered to produce a single-bit error, so that there are $m \cdot 2^m$ possible single-bit errors. To determine how many undetected single-bit errors exist, we must look at how to generate them.

Upon considering the various ways of generating single-bit errors that are undetectable, a few observations become obvious. We can never alter the final bit of a sequence, because that would change the transition count by plus or minus one, which would be detected. The only time we can alter a bit without getting caught is when a transition is adjacent to a double bit; that is, flipping the center bit in the patterns 001, 011, 100, or 110 will not affect the transition count. In other words, the transition count for ...0X1... and ...1X0... does not depend on the value of X.

Since our transition counter assumes an initial 0 state, the first bit of the sequence, regardless of its state, can be flipped without affecting the transition count, provided that the second bit is a one. In this case only the second of m bits is predetermined, i.e., $b_2 = 1$, and there are 2^{m-1} ways of completing the sequence. Any bit other than the first or last, that is, the $m-2$ bits from b_2 through b_{m-1}, can be altered without affecting the transition count if the bit in question is flanked by a zero on one side and a one on the other. For a given bit b_i we have free choice of $m-1$ bits, since as soon as we select b_{i-1} then b_{i+1} is forced to the opposite state. There are $(m-2) \cdot 2^{m-1}$ of these midstream errors. Adding the 2^{m-1} sequences where b_1 can be changed we have a total of $(m-1) \cdot 2^{m-1}$ sequences containing single-bit errors that cannot be detected by a transition counter. But earlier we showed that the total number of single-bit errors was $m \cdot 2^m$, hence the probability of failing to detect a single-bit error is

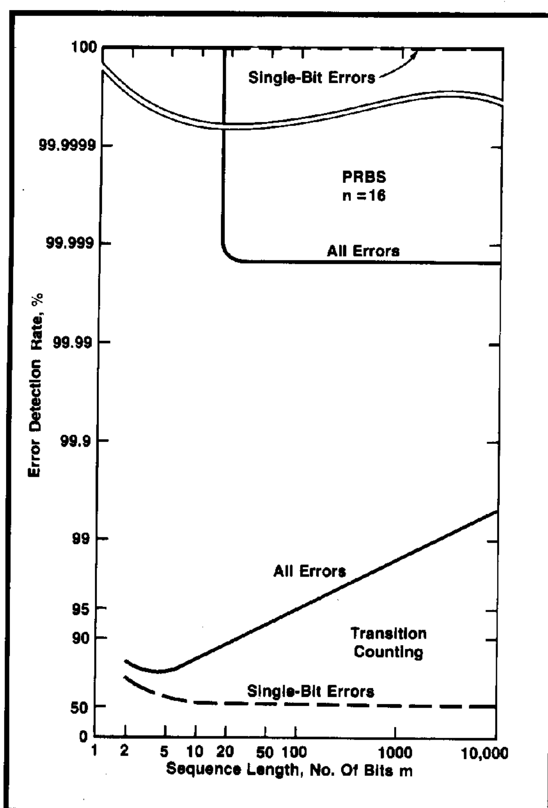


Fig. 5. *Probability of detecting errors for signature analysis and transition counting as a function of the length of the data sequence. n=16 for the PRBS generator.*

$$\text{Prob (Trnsn, Fail, single-bit)} = \frac{(m-1)\cdot 2^{m-1}}{m\cdot 2^m} = \frac{m-1}{2m} \approx 1/2.$$

It may be noted that the failure rate is actually somewhat higher, because a counter of limited length will overflow for long sequences, leaving some ambiguity. It can be shown that because of this overflow an n-bit transition counter will never detect more than $1/2^n$ of all errors.

Signature Analysis versus Transition Counting

We can now plot the probabilities of detecting any error using a transition counter versus a PRBS generator (see Fig. 5). It is interesting to note that the transition count method looks worst on single-bit errors, exactly where the feedback shift register never fails. Overall the transition counter looks pretty good, detecting at least half of all errors, but even a one-bit shift register could do that. The four-bit PRBS generator used in earlier examples will always detect better than $(100-100/2^4)=93\%$ of all errors. It seems conclusive that the PRBS method puts on a good performance, and if we want it to do better we merely add one more bit to the register to halve the rate of misses.

How Close Do We Want to Get?

We set out to find a means of instrument testing at least as good as present computer-based methods. These existing systems generally perform as well as the engineer who adapts them to the circuit under test. The task of adapting a circuit to be tested by signature analysis is very much the same as adapting to any other tester—engineering errors are assumed constant. If the PRBS technique is used for backtracing to find faulty components in field service, then the largest remaining block of human error is the ability of the service person to recognize a faulty signature.

It seems that a four-character signature is easily recognized, while the incidence of correct pattern recognition falls off with the addition of a fifth character. We tried this on a statistically small sample of people and found it to be so. Electronically, four hexadecimal characters is sixteen bits. A few bits more or less is not likely to complicate a shift register, but it would have an adverse effect on the user. Sixteen bits gives a detector failure rate of less than sixteen parts per million (one in 65,535), adequate for most purposes, so we settled on a four-character signature.

Since the signature offers no diagnostic information

Last In → A	B	C	D ← First In	Display
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9
0	1	0	1	A
1	1	0	1	C
0	0	1	1	F
1	0	1	1	H
0	1	1	1	P
1	1	1	1	U

Fig. 6. *In the HP 5004A Signature Analyzer, n=16 and the remainder, or signature, is displayed as four non-standard hexadecimal characters. Each character represents the outputs of a group of four flip-flops as shown here.*

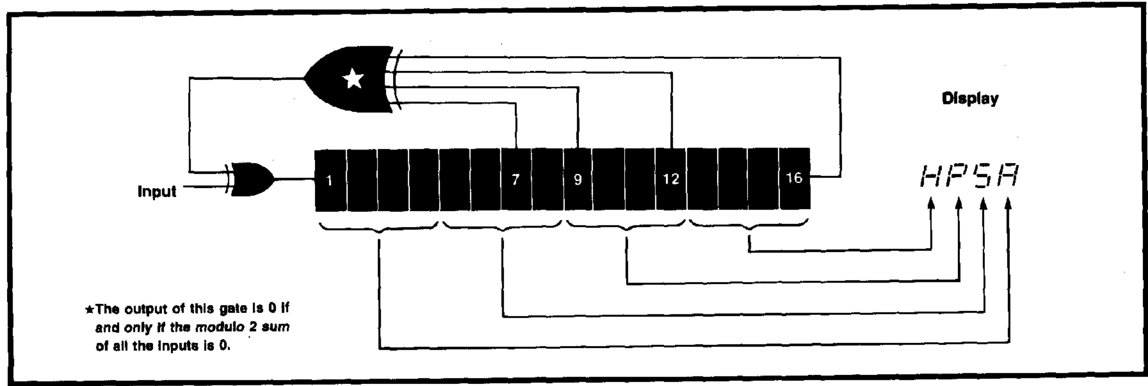


Fig. 7. *The 16-flip-flop PRBS generator used in the 5004A Signature Analyzer.*

whatsoever, but is purely go/no-go, the character set was not restricted, except to be readable. Numbers are quite readable but there are not enough of them. Another consideration was that for an inexpensive tool, seven-segment displays are desirable. The chosen character set (Fig. 6) is easily reproduced by a seven-segment display and the alpha characters are easily distinguishable even when read upside down. A further psychological advantage of this non-standard ("funny hex") character set is that it does not tempt the user to try to translate back to the binary residue in search of diagnostic information.

Register Polynomial

We have decided on a four-character display for a sixteen-bit register, but it remains to select the feedback taps to guarantee a maximal length sequence. It happens that this can be done in any of 2048 ways.[2] The computer industry uses two:

$$\text{CRC-16} = X^{16}+X^{15}+X^{2}+1,$$

and

$$\text{SDLC(or CCITT-16)} = X^{16}+X^{12}+X^{5}+1.$$

But each of these is reducible:

$$\text{CRC} = (X+1)\,(X^{15}+X+1),$$

and

$$\text{SDLC} = (X+1)\,(X^{15}+X^{14}+X^{13}+X^{12}+X^{4}+X^{3}+X^{2}+X+1).$$

The X+1 factor was included in both to act as a parity check; it means that all undetectable error sequences will have even parity. This is apparent by looking at the original polynomials and noting that they each have an even number of feedback taps, so an even number of error bits is required to cancel an error. For our purposes this clustering of undetectable errors seems undesirable. We would like a polynomial that scatters the missed errors as much as possible. For this reason we would also like to avoid selecting feedback taps that are evenly spaced or four or eight bits apart because the types of instruments, microprocessor-controlled, that we will most frequently be testing tend to repeat patterns at four and eight-bit intervals. The chosen feedback equation is:

$$X^{16}+X^{12}+X^{9}+X^{7}+1,$$

which corresponds to the characteristic polynomial

$$P(X) = X^{16}+X^{9}+X^{7}+X^{4}+1.$$

This is an irreducible maximal length generator with taps spaced unevenly (see Fig. 7). Our relatively limited experience with this PRBS generator has shown no problems with regard to the selection of feedback taps. The test of time will tell; even the CRC-16 generator seems to have fallen out of favor with respect to that of SDLC after having served the large-computer industry for well over a decade.

References

1. W.W. Peterson, and E.J. Weldon, Jr., "Error-Correcting Codes," The MIT Press, Cambridge, Massachusetts, 1972.
2. S.W. Golomb, "Shift-Register Sequences," Holden-Day, Inc., San Francisco, 1967.

Robert A. Frohwerk

Bob Frohwerk did the theoretical work on signature analysis. He received his BS degree in engineering from California Institute of Technology in 1970, and joined HP in 1973 with experience as a test engineer for a semiconductor manufacturer and as a test supervisor and quality assurance engineer/manager for an audio tape recorder firm. He's a member of the Audio Engineering Society. Bob was born in Portland, Oregon. Having recently bought a home in Los Altos, California, he and his wife are busy setting up a workshop for Bob's woodworking and metal sculpture, putting in a large organic garden, and planning furniture projects and a solar heating system. Bob also goes in for nature photography, sound systems, and meteorology (he built his wife's weather station).

Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits

It's a new tool for field troubleshooting of logic circuits to the component level.

by Anthony Y. Chan

THE NEW HEWLETT-PACKARD Model 5004A Signature Analyzer (Fig. 1) was designed to meet the need for field troubleshooting of digital circuits to the component level. The basic design goal was to implement the signature analysis technique described in the preceding article in a compact, portable instrument with inputs compatible with the commonly used logic families (TTL and 5V CMOS).

The 5004A is a service tool. It receives signals from the circuit under test, compresses them, and displays the result in the form of digital signatures associated with data nodes in the circuit under test. The signature analyzer does not generate any operational signal for circuit stimulus, depending instead on the circuit being tested to have built-in stimulus capability. The analyzer is capable of detecting intermittent faults. Its built-in self-test function increases user confidence and its diagnostic routine allows quick, easy troubleshooting with another 5004A if the instrument fails.

The signature analyzer's data probe is also a logic probe similar to the HP 545A Logic Probe.[1] The lamp at the probe tip turns bright for a logic 1, turns off for a logic 0, and goes dim when the input is open-circuited or at a bad level (third state).

What's Inside

Fig. 2 is a block diagram of the signature analyzer. During normal operation, the level detectors receive trains of start, stop, and clock control signals from the circuit under test and transmit them to the edge select switch. The edge select switch allows the user to

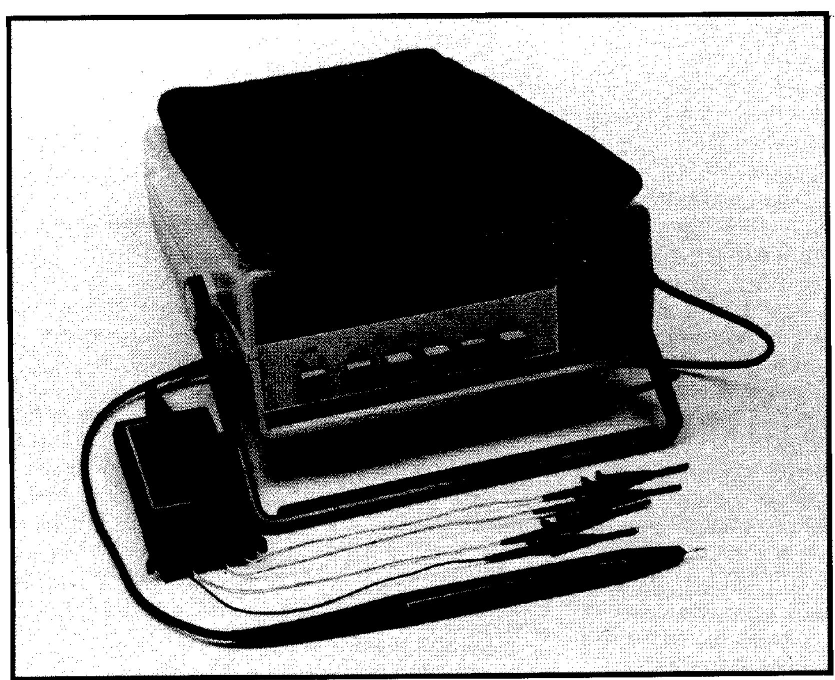


Fig. 1. *Model 5004A Signature Analyzer is a new tool for field-troubleshooting of digital circuits to the component level. (The circuits must be designed for signature analysis and must have built-in stimulus.) The 5004A gets start, stop, and clock inputs via its pod, shown here in the foreground, and data inputs via its data probe, which doubles as a logic probe.*

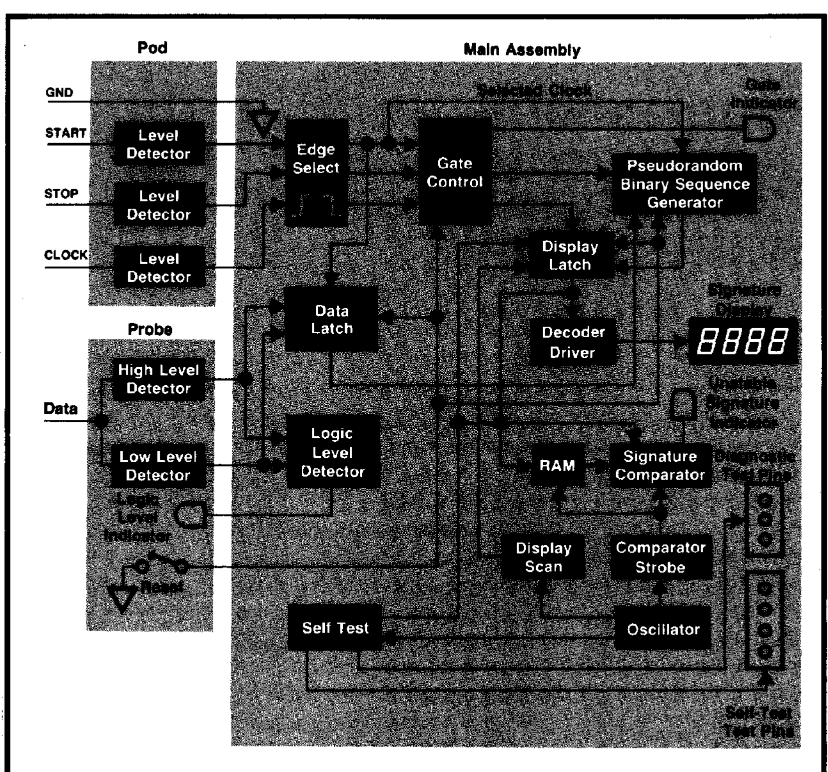


Fig. 2. *Model 5004A Signature Analyzer block diagram. The last 16 bits remaining in the PRBS generator when the stop signal occurs are loaded into the display latch and displayed as four hexadecimal characters.*

choose the polarity of signal transitions that the instrument will respond to. The gate control receives the selected control signals from the edge select switch and generates a gated measurement window (gate on) for the pseudorandom generator; it also turns the gate light on. The measurement window is the period between valid start and stop signals, and its length is measured in clock cycles (see Fig. 3). The minimum possible window length is one clock cycle.

The data probe translates voltages measured at circuit nodes into three logic states (logic 1, logic 0, and bad state) and transfers them to the data latch. The latch further translates the data, from three logic levels to two, at selected clock edges.* At each clock time, the data latch will pass a 1 or 0 level but will remain latched to the previous state if the input is in the bad state. The data latch may be the end of the road for some data because the pseudorandom generator accepts data only during the measurement window (gate on). Once data enters the pseudorandom generator, it is shifted in synchronism with the clock until the end of the measurement window. The last 16 bits remaining in the generator at gate-off time are loaded into the display latch and then output in the form of four non-standard hexadecimal characters—the signature. The display latch keeps the signature on until the end of the next measurement window, when the display is updated with new information. The signature will be stable as long as the measurement window and the data received within the window are repeatable.

*The probe recognizes three logic states instead of only two because of its logic-probe function.

Importance of Setup and Hold Times

Frequency response is one of the most important parameters in a test instrument. In the case of the

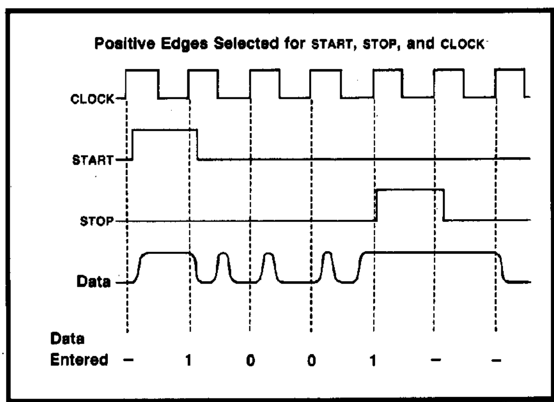


Fig. 3. *The measurement window is an integral number of clock cycles. One cycle is the minimum length.*

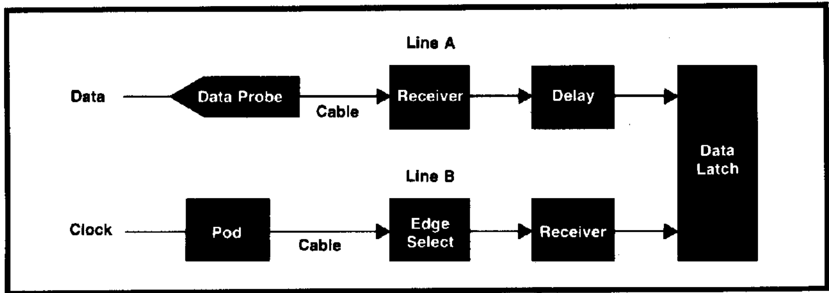


Fig. 4. *Delays through probe and pod channels are matched so that hold time (the time that input data must remain stable after a clock edge) is non-positive. Setup time (the time that input data must be stable before a clock edge) is typically 7 ns.*

signature analyzer, two other factors, data setup time and hold time, are very important as well. Data setup time is the interval for which data must be stable before the selected clock edge occurs. Hold time is the interval for which data must remain stable after the selected clock edge. Assuming a signature analyzer requiring 30 ns setup time and 10 ns hold time is used to test a circuit, then the logic of the circuit under test must be stable for at least 30 ns before the active clock edge and the logic must remain stable for 10 ns after the clock edge; otherwise, ambiguous readings may result. The setup and hold times limit the speed of the analyzer.

It is not easy for a high-speed circuit to guarantee that its logic will remain stable for some period of time after every active clock edge. The 5004A design goal was to be able to operate with reasonably short data setup time and non-positive hold time to minimize ambiguities.

Data and clock signals are received and transmitted to the data latch through the data probe, receiver, edge select switch, and cables (Fig. 4). There is one time delay for the data signal going through the data probe, cable, and receivers (line A), and another time delay for the clock pulses going through the wire and edge select switch into the data latch (line B). Every component, and therefore the time delays, may differ from unit to unit because of manufacturing tolerances. To guarantee a non-positive hold time, eliminate race conditions, and be reproducible in a production environment, the minimum delay of line A must be equal to or longer than the maximum delay of line B ($t_{Amin} \geqslant t_{Bmax}$). Also desired is a minimum setup time $T_s = t_{Amax} - t_{Bmin}$.

One way to achieve short setup time is to have identical circuitry in the data and clock channels, so propagation delays cancel each other. Circuitry in the data probe is very similar to that in the pod. The receivers in both channels are identical and share the same IC chip. The edge select switch in the clock line has very little delay. The symmetry results in a good match between the two signal lines, but to insure that $t_{Amin} \geqslant t_{Bmax}$, there is a delay circuit in line A, and the cable length of line B is shorter. Thus it is possible to guarantee hold time less than 0 ns and setup time less than 15 ns (7 ns typical).

Input Impedance

Since the 5004A is a test instrument, it is important that its inputs do not load or condition the circuit under test. It is generally true that high input impedance reduces loading. But, how high can the input impedance be before other effects cause problems?

Let's study a few cases of high-impedance input in a synchronous device (Fig. 5). Fig. 5a shows the result of the input data's changing from a logic 1 to a third state at clock 1. The input voltage is pulled toward ground by the pull-down resistance. The difference between the solid line and the dotted line is the difference of node capacitance (C) and pull-down resistance (R) tolerances. Depending on the clock rate and the RC difference, the result at clock 2 can be either a logic 0 or a third state. The same thing in reverse happens in case B. In case C, the input data is changing to an intermediate level (~1.4V). When the data changes from a logic 1 to the third state, the input is pulled towards 1.4V. The result at clock 2 is in the third state no matter what the RC time constant is.

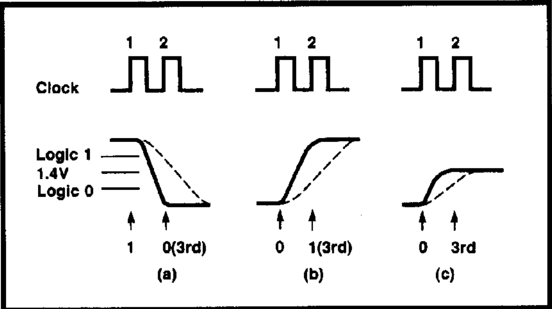


Fig. 5. *5004A input impedance is returned to 1.4V to eliminate ambiguities caused by input RC tolerances. Here are three possible results of the input data's changing from a 0 or 1 at clock 1 to a third state at clock 2. The solid and dotted lines are for different values of input RC. In (a), the input at clock 2 can be seen as a logic zero or a third state, depending on the value of RC. (b) is the reverse of (a). In (c), with the input returned to 1.4V, the result is always a third state.*

There are two other advantages to returning the input impedance to 1.4V. First, there is less voltage swing, and almost equal swings for both logic 1 and logic 0 states. Second, the logic probe open-input requirement is met.

Very high input impedance may cause problems even for a non-clocked device. It introduces threshold errors because of the offset bias current of the input amplifier, and the leakage current of the three-state bus might change the measured voltage. After study and calculation, we chose 50 kΩ to 1.4V as the input impedance and return voltage for the 5004A inputs. 50 kΩ is large enough not to load TTL and most 5V CMOS logic families, and small enough not to cause excessive offset voltage with typical leakage currents on a three-state bus.

Construction

The 5004A Signature Analyzer is constructed in a lightweight, rugged case. A hand-held data probe and a small rectangular pod are connected to the instrument by cables (Fig. 1). Inside the main case are the edge select circuit, gate control, data latch, pseudorandom generator, display latch, signature displays, signature comparator, self-test stimulus generator, and power supply shown in Fig. 2. All the electronics and mechanical components are mounted on a single printed circuit board assembly sandwiched inside two shells held together with four screws. On the front panel are four large seven-segment displays. A light to the left of the display shows gate (measurement window) activity while one on the right indicates unstable signature. Six pushbutton switches control power on/off, start, stop, and clock edge polarities, a hold mode for single cycle events or freezing the signature, and a self-test mode. Start, stop, clock, and data test sockets on the right-hand side of the front panel are for self-test and diagnostic setup. A soft pouch mounted on top of the instrument stores the data probe, pod, and necessary accessories when not in use.

Data Probe

The active data probe is a hand-held probe. Its main function is to accept tip logic information with minimum tip capacitance. The input signal is connected to two comparators through voltage dividers and an RC network (see Fig. 6). The voltage divider R1-R2 is terminated at 1.4V, which guarantees an open input at a bad level and eliminates the potential ambiguity, discussed earlier, resulting from RC tolerances.

Input overload protection is provided by on-chip clamp diodes and the external network R1 and CR1. C1 provides a bypass for fast transitions. R3, R5, and R6 set up voltage references for comparators A and B. Two comparators are needed to measure the three logic states—1, 0, and bad level. The high-speed differential-in/differential-out comparators translate the input voltage into digital signals and transmit them to the main instrument through twisted pairs. A single-contact pushbutton switch on the data probe resets the pseudorandom generator, state control, and displays.

Pod

The thin, rectangular pod houses three identical

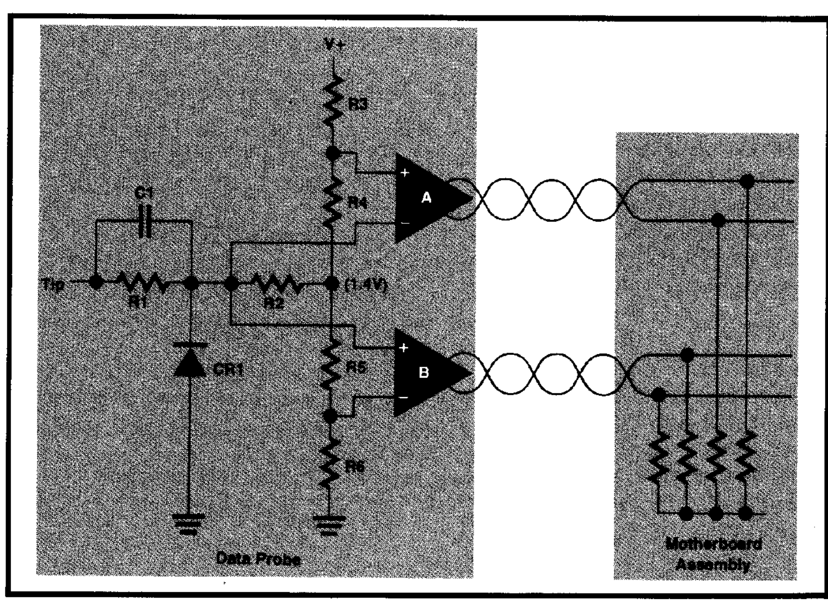


Fig. 6. *Simplified 5004A data probe schematic.*

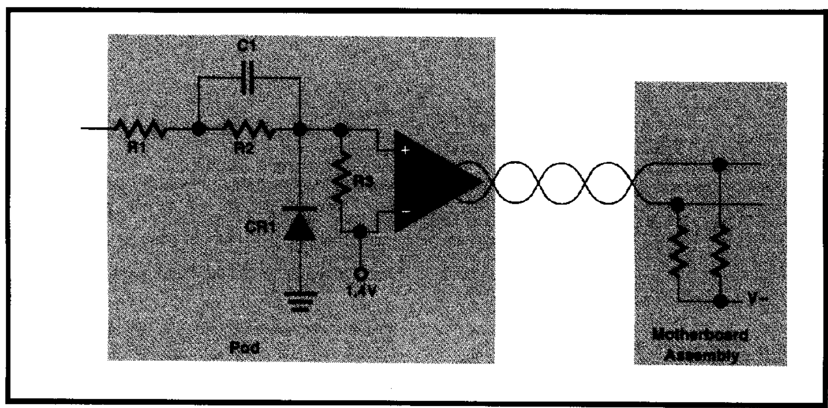


Fig. 7. *Simplified schematic of one channel of the 5004A input pod.*

channels for start, stop, and clock control inputs. The input wires can be directly plugged into any 0.03-inch round socket or connected to a "grabber" that can hook onto almost any test pin and is particularly good for IC pins. Each of the control channels is very similar to the circuitry in the data probe to match propagation delays. In fact, delay match is the major function of the pod.

In each channel of the pod is a comparator of the same type as those in the data probe. One of the comparator inputs is connected to voltage divider R1-R2-R3 while the other input is connected to 1.4V along with R3; this increases input hysteresis and sets the input threshold (Fig. 7). The impedance of R1-R2-R3 is the same as the impedance of the data probe (50 kΩ); termination at the same voltage level further improves matching.

Protection against input overload is provided by internal clamp diodes in the comparator IC and by external network R1, R2, and CR1. R1 damps the ringing generated by the inductance of the input wire. C1 provides a bypass for fast transitions.

Unstable Signature

An intermittent fault is one of the biggest problems in electronic repair. The fault comes and goes, and in most cases does not stay long enough for positive detection. Signature analysis can detect such faults if they occur within a measurement window. However, the operator may not receive the message if the measurement cycle time is too short.

The random-access memory (RAM) in the main assembly of the 5004A continuously writes and reads the display information from the display latch at the display scan rate. During each scan cycle, the signature comparator compares the signature stored in the RAM with the one in the display latch, and turns on the unstable signature light on the front panel when any difference exists. This light is stretched for 100 ms to allow recognition. The comparison is done on a sampled basis and not each time a new signature is developed, so the unstable signature detector works most of the time, but not 100%. Errors occurring in a very short measurement cycle may not always be detected by the relatively slow-scanning comparator.

Hold Mode

The hold mode works closely with the stop signal. If the hold switch on the front panel is pushed in, the hold mode will be entered at the end of the measurement window, freezing the signature display and preventing the gate control from starting a new cycle. Hold mode is particularly useful for testing single-shot events like the start-up sequence of a system.

Self Test

It is important for a user to know that a test instrument is in good working condition before it is used to test anything. The 5004A has a built-in self-test function that gives a quick, accurate check of the instrument. Pressing the self-test switch on the front panel energizes the self-test ROM, which interrupts the display update and generates a special programmed stimulus of start, stop, clock, and data signals to the test sockets on the front panel. With the start, stop, and clock control inputs connected to the corresponding test sockets and the data probe to the data test socket on the front panel, and with positive edges selected for the start, stop, and clock inputs, the signature analyzer performs the self test. When a good working 5004A is tested, its gate light flashes, the unstable signature light blinks, the logic light at the data probe tip flashes, and the signature displays 3951, 2P61, 8888 and then repeats. Pushing the hold switch in turns the gate light off and the signature displays 8888, 3951, or 2P61. The self-test routine tests the entire instrument except the clock edge select circuit and the ground wire at the pod input.

Fig. 8. *The 5004A Signature Analyzer is designed for troubleshooting with another 5004A should the self test reveal a fault. Sliding the NORM/SERVICE switch to the SERVICE position opens the three feedback loops in the instrument.*

Diagnostic Routine

When an unexpected result during self test indicates a fault, troubleshooting and repair are required. The 5004A was designed with signature analysis in mind. It can be tested with another 5004A. The instrument's top cover can be easily removed by removing four hold-down screws on the bottom and loosening two heat sink mounting screws on the back. All the components in the instrument's main case are then exposed for testing. The failing instrument is placed in self-test mode and the start, stop, clock, and ground inputs of a known good 5004A are connected to the test sockets located on the left side of the printed circuit board in the main case. Probing the circuit nodes with the data probe, reading the signatures on the analyzing 5004A, and comparing them with those printed on the schematic is an easy and almost error-free way of determining the quality of a circuit node. Once a faulty node is found, the source of the problem can be easily located with standard backtracing techniques.

When the fault is in a feedback loop, any single fault will cause all the nodes within the loop to appear bad. To pinpoint the fault, the loop must be opened. There are three feedback loops in the signature analyzer, and a slide switch (NORM/SERVICE) on the left of the main printed circuit board is provided for opening them (Fig. 8). Sliding the switch to the SERVICE position opens all three loops.

The diagnostic routine works on the entire instrument except the power supply, the ECL circuits in the data probe and pod, and their interface circuits.

SPECIFICATIONS
HP Model 5004A Signature Analyzer

DISPLAY:
SIGNATURE: Four-digit hexadecimal.
Characters 0,1,2,3,4,5,6,7,8,9,A,C,F,H,P,U.
GATE UNSTABLE INDICATORS: Panel lights. Stretching: 100 ms.
PROBE-TIP INDICATOR: Light indicates high, low, bad-level, and pulsing states. Minimum pulse width: 10 ns. Stretching: 50 ms.

PROBABILITY OF CLASSIFYING CORRECT DATA STREAM AS CORRECT: 100%.

PROBABILITY OF CLASSIFYING FAULTY DATA STREAM AS FAULTY: 99.998%.

MINIMUM GATE LENGTH: 1 clock cycle.

MINIMUM TIMING BETWEEN GATES (from last STOP to next START): 1 clock cycle.

DATA PROBE:
INPUT IMPEDANCE:
50 kΩ to 1.4V, nominal. Shunted by 7 pF, nominal.
THRESHOLD:
Logic one: 2.0V −.1, −.3
Logic zero: 0.8V +.3, −.2
SETUP TIME: 15 ns, with 0.1V over-drive. (Data to be valid at least 15 ns before selected clock edge.)
HOLD TIME: 0 ns (Data to be held until occurrence of selected clock edge.)

GATING INPUT LINES:
START, STOP, CLOCK INPUTS:
Input Impedance: 50 kΩ to 1.4V, nominal. Shunted by 7 pF, nominal.
Threshold: 1.4V ±.6 (.2V hysteresis, typical).

START, STOP INPUTS:
SETUP TIME: 25 ns. (START, STOP to be valid at least 25 ns before selected clock edge.)
HOLD TIME: 0 ns. (START, STOP to be held until occurrence of selected clock edge.

CLOCK INPUT:
MAXIMUM CLOCK FREQUENCY: 10 MHz.
MINIMUM CLOCK TIME IN HIGH OR LOW STATE: 50 ns.

OVERLOAD PROTECTION: All inputs ±150V continuous, ±250V intermittent, 250Vac for 1 min.

OPERATING ENVIRONMENT: Temperature: 0-55°C; Humidity: 95% RH at 40°C; Altitude: 4,600 m.

POWER REQUIREMENTS: 100/ 120 Vac, +5%, −10%, 48-440 Hz.
220/ 240 Vac, +5% −10%, 48-66 Hz.
15 VA Max.

WEIGHT: Net: 2.5 kg, 5.5 lbs. Shipping: 7.7 kg, 17 lbs.

OVERALL DIMENSIONS: 90 mm high × 215 mm wide × 300 mm deep (3½ in × 5½ in × 12 in). Dimensions exclude tilt bale, probes, and pouch.

PRICE IN U.S.A.: $990.

MANUFACTURING DIVISION: SANTA CLARA DIVISION
5301 Stevens Creek Boulevard
Santa Clara, California 95050 U.S.A.

Anthony Y. Chan

Tony Chan received his BSEE degree from the University of California at Berkeley in 1969 and his MSEE degree from California State University at San Jose in 1974. With HP since 1973, he developed the IC chip for the 546A Logic Pulser and designed the 5004A Signature Analyzer. Before joining HP, he designed linear and digital ICs for four years. A native of Hong Kong, Tony now lives in Sunnyvale, California. He's married and has two children. Having just finished remodeling his home, he's now taking on landscaping and furniture-building projects. He likes working with his hands, especially on wood and automobiles, and enjoys an occasional game of tennis.

Signature Analysis—Concepts, Examples, and Guidelines

Guidelines for the designer are developed based on experience in attempting to retrofit existing products for signature analysis and the successful application of signature analysis in a new voltmeter.

by Hans J. Nadig

THE POWER OF SIGNATURE ANALYSIS as a field troubleshooting technique is amply demonstrated by the analysis presented in the article on page 2 of this issue. The technique can even pinpoint the 20% or so of failures that are "soft" and therefore difficult to find, taking 70-80% of troubleshooting and repair time. Soft failures include those that occur only at certain temperatures or vibration levels. They may be related to noise performance or marginal design, such as race conditions that occur only when the power supply voltage is low but still within specifications. Or they may occur only when the user gives the machine a certain sequence of commands.

Signature analysis is applicable to complex instruments using microprocessors and high-speed algorithmic state machines. Yet it is simple enough so that the user of a product may be able to apply it nearly as well as more highly trained field service personnel.

Having recognized the power of signature analysis, we first attempted to apply it to existing products, including computers, CRT terminals, and the digital portions of microwave test equipment. We soon recognized that either the circuits had to be altered or the signature analysis approach would be no better than earlier methods. After some experience we were able to define rules for making a product compatible with signature analysis. These rules, summarized on page 18, are guidelines for the designer. Following them helps assure that a product will be simple and inexpensive to troubleshoot by the signature analysis method.

How We Got Started

A good way to demonstrate the advantages of signature analysis and the requirements for applying it successfully is to describe what happened when we first tried it a few years ago.

With a prototype signature analyzer we set out to apply the technique to various Hewlett-Packard instruments. We first attacked a CRT terminal with microprocessor control and ROM and RAM storage, including some dynamic memories.

The built-in self-test mode of the terminal displayed a certain test pattern on the CRT and flashed the cursor at a 2½-Hz rate. Taking advantage of this self test as a stimulus, and using the most significant address bit to start and stop the measurement, we soon recognized that these signals did not provide a stable measurement window. Some portions of the terminal operated on an interrupt basis, so the number of clock periods varied within the START-STOP window. Needless to say, the data stream changed, too. Next we concentrated our efforts on one section, the memory. To test it, we wanted to force the microprocessor into a mode in which all the memory locations were addressed, but to do this, we were forced to cut the data bus. Fortunately, we could separate the microprocessor from the data bus by using an extender board and cutting the lines there. Grounding a few lines and pulling some other lines high caused the microprocessor to repeatedly execute one instruction that automatically incremented the address each cycle, effectively stepping up through the whole address field. Fig 1 shows how this can be done for an 8080 microprocessor.

At this point we realized that the microprocessor, the clock, and the power supply were the heart of the product. We decided to call this the "kernel" (Fig. 2). By verifying the proper operation of these parts first we could then expand and test additional portions of the circuitry. With the free-running microprocessor exercising the control and address lines, we were able to test the address bus, the ROMs, and the data bus. As a START and STOP signal the most significant bit of the address bus was used, allowing us to check all the ROM locations. Since a number of RAMs were also affected, we applied a grounded jumper wire to force the enable line to the RAMs low; this was necessary to get stable signatures, since the RAMs did not contain a defined data pattern, and if addressed, randomly altered the information on the data bus.

Here, then, were our first lessons: *feedback loops cause problems unless opened; circuits not related to*

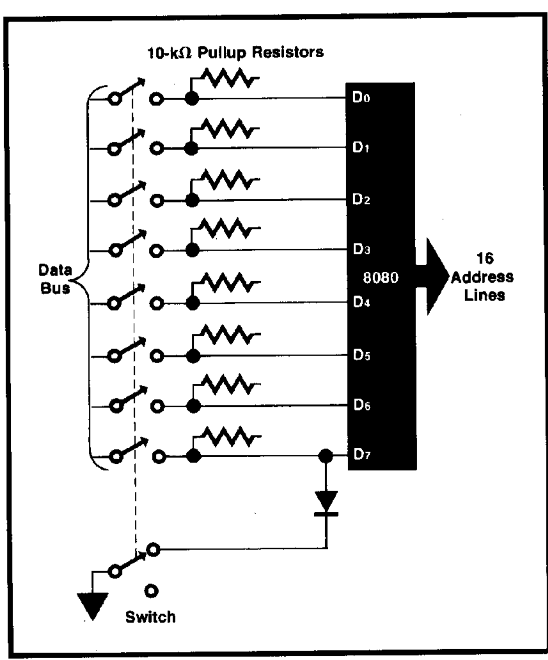


Fig. 1. *To be an effective troubleshooting technique, signature analysis must be designed into a product. For example, for a test of the address lines of a microprocessor, there should be a switch that opens the data bus and forces the microprocessor to free run. The address lines can then be checked and can also be used as control inputs to the signature analyzer.*

the test must be disabled.

Synchronous Operation Necessary

It would be ideal if one setup would allow troubleshooting most parts of an instrument. The synchronization signal with the highest rate would be connected to the clock input of the signature analyzer.

Our display terminal uses a number of different frequencies, from 21 MHz down to 1¼ Hz. A ripple counter divides the frequencies down. Trying to characterize the divider chain showed us unstable signatures for every node after the first stage. The reason was that the circuits operated asynchronously with as much as 500 ns skew from the first to the last stage. Lowering the clock frequency to about 2 MHz by removing the crystal from the oscillator, we were able to define stable signatures for the counter chain. However, one measurement lasted 10 seconds, and to verify whether it was stable or not we had to have at least two complete measurements. An alternative to reducing the clock frequency was a new test setup for the slower parts of the divider. However, it is always wise to minimize the number of necessary setups.

So we learned that *synchronous operation is essential for high-speed testing.* Fortunately, this is easy to accomplish in most microprocessor designs, even those with the newer types of microprocessors that use asynchronous handshake lines to gate information in and out. Although it might seem at first that signature analysis is not applicable in such a case, the problem of asynchronous operation can be eliminated if the handshake lines are used to clock the data into the analyzer. Also, when this is done third-state conditions are no longer a problem because at the time of the "data valid" signal the data is either high or low. Thus a seemingly asynchronous system can behave as a synchronous system as seen by the troubleshooting tool, the signature analyzer.

Need for Designed-In Capability

An interesting possibility is that of measuring all the possible fault conditions at a central node by inducing faults into a good circuit and recording the corresponding signatures at the central node in a signature fault table. Testing the central node then tells the whole story of whether the instrument is in working condition or not. If it fails, the fault table indicates where the error is and sometimes even which part has to be replaced.

In the case of the terminal, an ideal central node seemed to be the video signal. Every data and control line is ultimately concentrated in one node containing all the information to scan a dot across the screen. Using the terminal's self-test feature as a stimulus, we chose the new-frame trigger signal as our START and STOP inputs. But for some reason we could not get stable signatures, which meant that the data stream between the two gate signals was not the same for each frame.

The culprits were two signals that occurred at a much slower rate than the 60 Hz for the frames. The blinking signals for the 2½-Hz cursor and for the 1¼-Hz enhancement were changing the characteristic data stream for the frames. Not until we disabled the signal generators for the blinking did we get stable signatures.

If parts of the circuit are being disabled the comprehensiveness of the test is reduced. In this case the designer could have provided the necessary setup to do a complete test. But after the design is frozen without signature analysis in mind it is hard to apply it successfully. If the window for signature analysis is selected so that the slowest blinker is the trigger for START and STOP, it is possible to create a stable signature or, in other words, a repeatable data stream.

The characterization of the RAM required special attention. A defined pattern had to be loaded into the memories before useful signatures were obtained at the outputs. By using several jumpers to enable the write cycle and the ROM outputs, then switching into

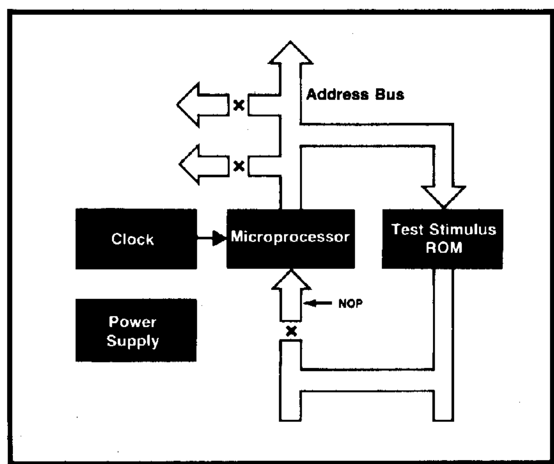


Fig. 2. *Signature analysis test procedures should verify the operation of key portions—the "kernel"—of a product first, then use the kernel to test other circuits. A typical kernel might consist of microprocessor, clock, power supply, and one or more read-only memories.*

a read mode for the RAM and disabling the ROM, we effectively loaded the content of the ROMs into the RAMs. After that, valid readings were obtainable and it was possible to trace down a bad RAM component.

Having tested and characterized about 30% of the digital boards, we next concentrated our efforts on the large display memory. Testing this dynamic memory was not easy, because it went through an asynchronous refresh cycle every 2 ms. Even adding more jumper wires, we had to admit finally that without cutting leads or altering the circuit we would not get any satisfactory results.

Looking at the CRT terminal with the oscillator crystal removed, with a cut-up extender board and jumper wires clipped into the circuit here and there, we learned the most important lesson: *signature analysis capability has to be designed into the circuit.*

After that a number of additional products were tested and the message remained the same: retrofitting is not an effective approach. On the other hand, it became clear that the additional effort to make the circuit signature-analysis-compatible is indeed very small if done at an early stage of the product development. Early, in this case, means the breadboard stage.

Thoughts on Implementation

The versatility of the signature analysis concept is impressive. As long as the data is valid at the selected clock edge, and the stimulus is repeatable, many parameters can be selected. The window length, or the number of bits in the data stream, can be of any value (100,000 bits is not unusual). Any suitable signal can be selected as the clock input, enabling the designer to make seemingly asynchronous circuits look as if they were synchronous. A major advantage is that everything happens at normal speed.

The implementation of signature analysis into a product is similar to designing a microprocessor into a product. In the latter case, the designer has to learn the instructions, and has to understand the advantages and the limitations of the microprocessor. Because of the learning curve, the first application will most likely take more time than later designs. Also, there is no cookbook approach to a microprocessor design because there are no two situations alike. The designer makes decisions based on the evaluation of power consumption, cost, size, reliability, and so on.

The same is true for the implementation of signature analysis. The design engineer must understand the function of each component and create a test stimulus that tests each function totally. Simply exercising a node may not be enough. Even a component as simple as an AND gate may have stable and correct input and output signatures and still be bad, as shown in Fig. 3. Similar cautions apply to any test method, of course. The designer must be careful to *test completely the function of the smallest replaceable part*.

Serviceability is an additional algorithm. If the service algorithm is taken into consideration at an early stage of the development, the application will be easier, and the additional cost for hardware, test program memory space, and development time will be offset by shorter test times in production. Also, the warranty service and repair costs will be much lower. Later in an actual example, we will see how even the

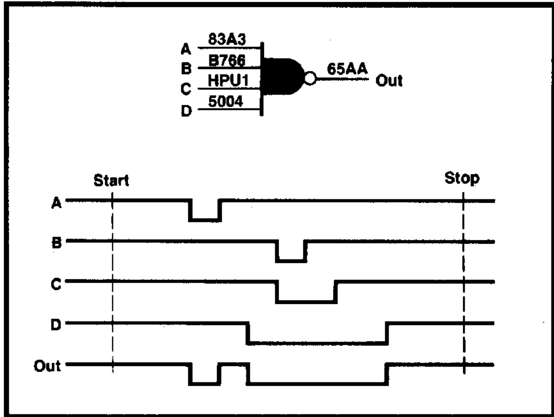


Fig. 3. *The test stimulus should test each component thoroughly. Otherwise a circuit, such as this AND gate, can have stable and correct signatures at each input and output node and still be bad. For example, input B or C might have an open bonding wire inside the IC, but in this case the error is masked by input D. Careful test stimulus design avoids this.*

Designer Guidelines for Applying Signature Analysis to Microprocessor-Based Products

General Guidelines

- Make a full commitment to use signature analysis at the definition stage of the product.
- Evaluate the trade-offs, such as increased factory costs versus lower test time in production and lower warranty and repair cost in the field. Other factors that might influence the decision are warranty cost goal, profit, cost of field repair, acceptable downtime, the cost of alternative service procedures like board exchange programs, the topography of the service organization, and the extra cost for customs if the parts have to be shipped back and forth across country borders.
- Familiarize yourself with the signature analysis service philosophy and allow some extra time for the design.
- Start to prove the basic working of signature analysis at the breadboard stage, before laying out printed circuit boards.
- Team up with the service engineer who will write the manual for the product. Do it at an early stage, before the first prototype is finished.
- If you hope for some benefit for production testing, get the production engineer involved during your definition of the test stimulus and the method of connecting the signature analyzer.
- As a design engineer, be aware that the volume of the necessary documentation can be minimized by selecting the appropriate partitioning of the tested sections in the product.

Technical Rules

- The stimulus for troubleshooting comes from within the product. The self-test stimulus can frequently be used.
- Provide if possible a free-running repeatable stimulus for continuous cycling.
- Tested nodes are to be in a valid and repeatable state at the time of the selected clock edge for triggering.
- Provide easy access to the START, STOP, and CLOCK test points.
- Feedback loops must be capable of being opened. Only an open-ended test allows backtracing.
- The test program or stimulus should exercise within the START-STOP window all the functions that are used in the instrument, although it is not necessary to perform a meaningful operation.
- Provide a controlled test stimulus to interrupt lines, open connectors, and signals that are normally asynchronous.

Additional Guidelines

- Verify the heart or kernel of the instrument first. The kernel may consist of the power supply, the clock generator, and the microprocessor. Then, use this central part to create the stimulus for the peripheral circuits.
- ROMs may be used to write the stimulus program.
- Divide the circuit into well defined portions. Several test setups may be necessary.
- Avoid the use of circuits with non-repeatable delays (e.g. one-shot multivibrators) within the test loop.
- Avoid, if possible, the third-state condition of a three-state node during the measurement cycle.

factory cost can be lowered in spite of needing some extra components, because the whole circuit could be placed on a single large board, while for the traditional board swap service approach, the circuit would have been divided into a number of easy-to-replace subassemblies, which would have required more connectors and hardware to hold the boards in place.

Signature Analysis and the Service Engineer

How would a service engineer use signature analysis if a product failed? The assumption is made that the signature analysis method is designed into the product. Instructions on the schematic or in the service manual show how to switch the product into the diagnostic mode and how to connect the signature analyzer to the device under test. Each node on the schematic is marked with a signature (Fig. 4). With the aid of the schematic the service engineer first reads the output signatures of the device under test. If they are bad, he traces back to a point in the circuit where a good signature appears at the input side of a component and a bad one at the output side. This is called backtracing.

Some understanding of the components in a digital schematic is essential. The direction of the data flow is important but no special knowledge about the actual function of the assembly is required. So, one advantage of the signature analysis service method is that less training is needed to learn to do fault tracing. We can even go a step further and develop a troubleshooting tree without the use of a schematic. A picture of the physical board with signatures at the pins of IC's or components may be used instead (Fig. 5). This way the technician is not required to know whether the circuit he is testing contains a complex storage device or simply a gate. One suggestion is to print the signatures onto the printed circuit board itself, with arrows indicating the signal flow. Another is to print a test template that is attached to the component side of the circuit board when service is required (see Fig. 6). Holes in the appropriate locations, signatures, and other instructions printed on the template guide the service person to the faulty node.

The 3455A Voltmeter—an Example

The first HP instrument using signature analysis is the 3455A Digital Voltmeter[1] (Fig. 7). The digital portion of this instrument is quite extensive. It is microprocessor controlled and contains an elaborate self-test program stored in ROM. If the self test fails, a jumper inside the enclosure is removed, breaking feedback loops and enabling the self-test program, which is then used to troubleshoot the instrument.

Signature analysis influenced other factors that make this voltmeter easier to troubleshoot down to the component level. The entire digital portion is on a

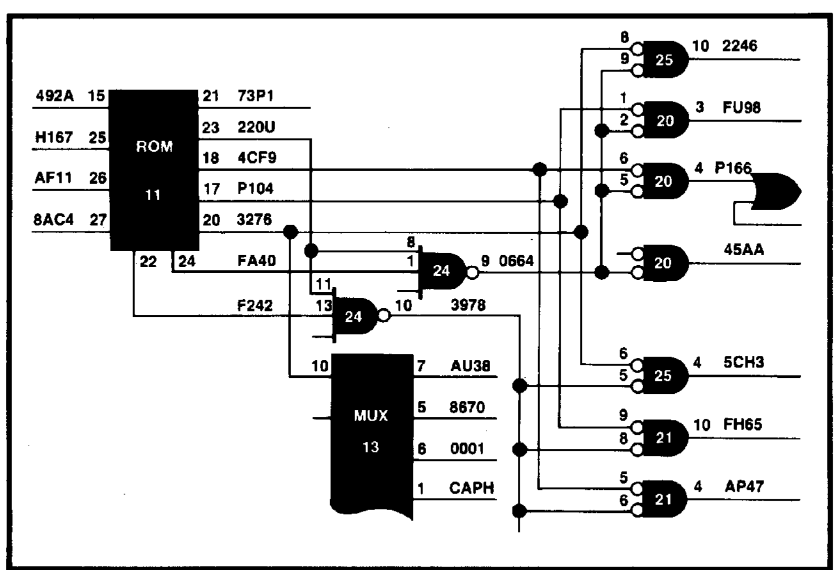


Fig. 4. *An example of an annotated schematic, showing correct signatures at various circuit nodes.*

single board. The elimination of connectors and a multitude of smaller subassemblies not only reduced production cost, but also made all the parts easily accessible for testing without special extender boards.

Some extra design time, a few more ROM locations, and the extra jumper wire were the price paid for serviceability. A cost evaluation verified that the production cost was lowered. The extra design time amounted to approximately 1% of the overall development time.

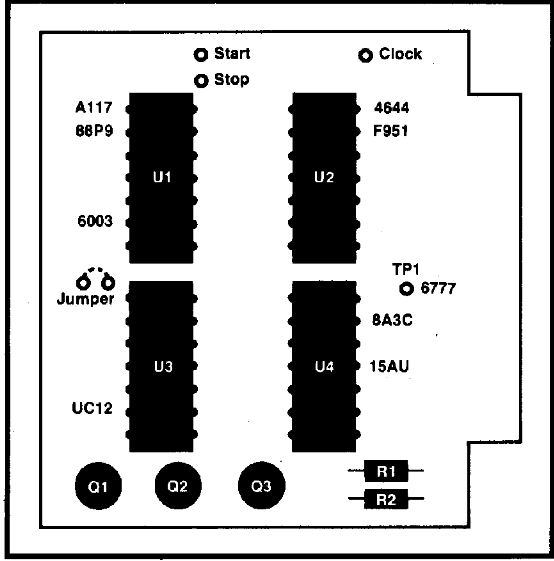


Fig. 5. *A service manual may use a picture or drawing of the board being tested, showing proper signatures at various test points. A troubleshooting tree in the manual guides the service person, who need not know the function of each component. A board overlay or template may also be used.*

Besides the design engineer, the service engineer who wrote the service manual made an important contribution to the successful application of signature analysis. He learned the internal algorithms of the product almost as well as the designer. Because there was no precedent to fall back on, he used a number of innovative ideas, which have been well accepted by the field engineers.

The service manual guides the service person to the fault within a very short time. The manual contains a troubleshooting tree that, combined with annotated schematics and graphs of board layouts, leads directly to the bad node. In some cases the manual gives instructions as to which IC to replace. In other cases the use of a logic probe, which may be the 5004A Signature Analyzer's data probe, may be required. A current sensor helps to find short circuits between traces or to ground and is particularly helpful if a long bus line should fail. A portion of the 3455A Voltmeter troubleshooting tree is shown in Fig. 8.

The first test checks the kernel, which consists of the microprocessor, the clock circuit, the power supply, and a number of external gates. After proper functioning of the kernel is verified, the test setup is changed (one control input of the signature analyzer is moved to another pin) and the remaining portions of the circuit are tested.

A special portion of the ROM control loads and reads the RAMs. Some asynchronous portions require a third test setup. Again, the connection of the START wire is simply moved to the next pin designated for

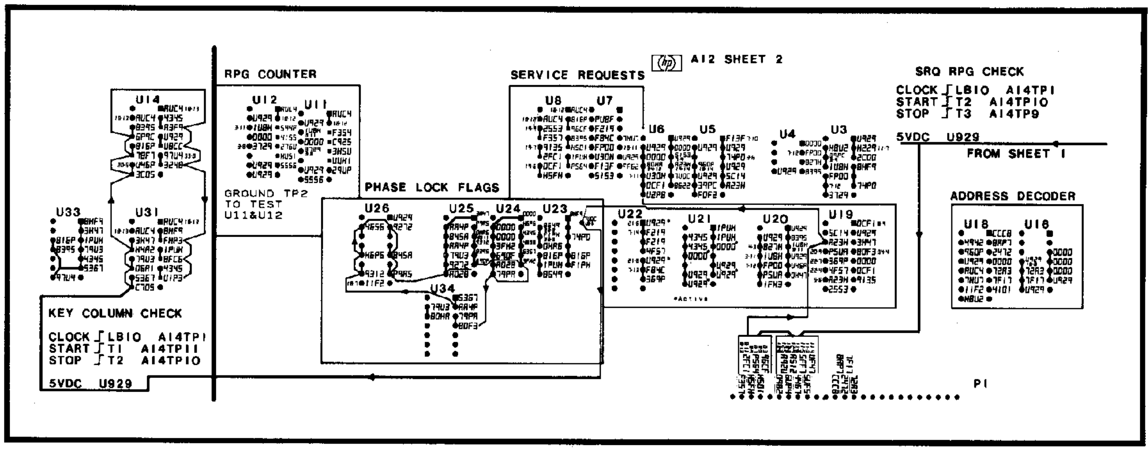


Fig. 6. *A template for signature analysis troubleshooting. The template is attached to the component side of the circuit board. Holes allow probe access to the test points. If the test point is not a source, the origin of the signal (IC and pin number) is listed next to the correct signature.*

this purpose and troubleshooting can continue.

It is obvious that proper documentation is essential. The 3455A manual shows, for each test setup, a picture of the board. Only the signatures related to that particular test are given. This helps to direct the effort towards the important areas on the board. Interrupt signals are simulated by the ROM program so they occur repeatedly at the same spot within a window and stable signatures result.

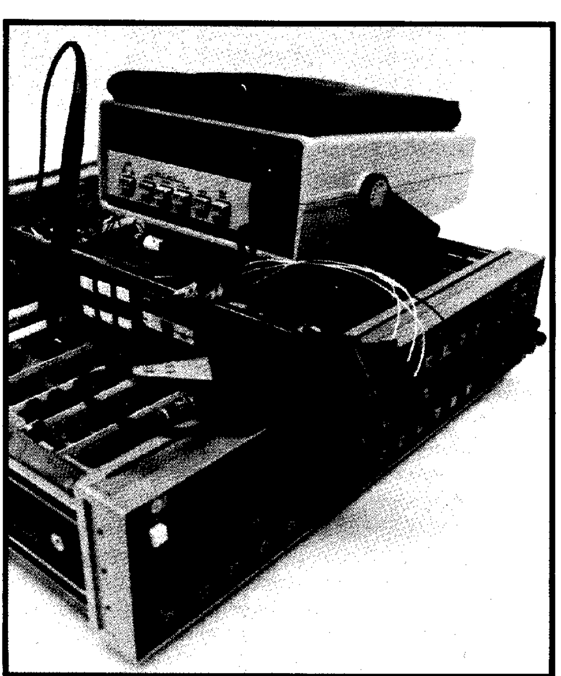


Fig. 7. *Model 3455A System Voltmeter is the first HP instrument designed for troubleshooting with the 5004A Signature Analyzer.*

When all the signatures seem to be bad, the question arises whether the test setup itself is correct. The 5004A Signature Analyzer's self-test feature can be used to check it for proper operation. Each test setup can then be tested by touching V_{cc} with the 5004A probe. If this characteristic signature is correct, it means that the START and STOP channels are triggered at the correct moments and that the number of clock pulses within the measurement window is correct. It also tells the user that the switches on the signature analyzer are set correctly, and that all the jumpers, switches, and control buttons in the voltmeter are set to the right position. Thus the confidence level is very high at the beginning of a test routine.

A conclusion drawn from this application is as follows: success is assured if the service engineer works closely with the design engineer. This also saves time at the end of the development phase because the service engineer is fully aware of the new product's internal operation. It also forces the designer to think about serviceability.

The fact that signature analysis is built into the 3455A Voltmeter not only made serviceability but also final testing on the production line much easier. The signature analyzer is now a standard piece of equipment on the production line.

Acknowledgments

The 5004A Signature Analyzer is a result of the effort of many engineers in HP's Colorado Springs and Santa Clara Divisions. Loveland Instrument Division and Santa Rosa Division added a great deal to definition of the needs and conditions of the new

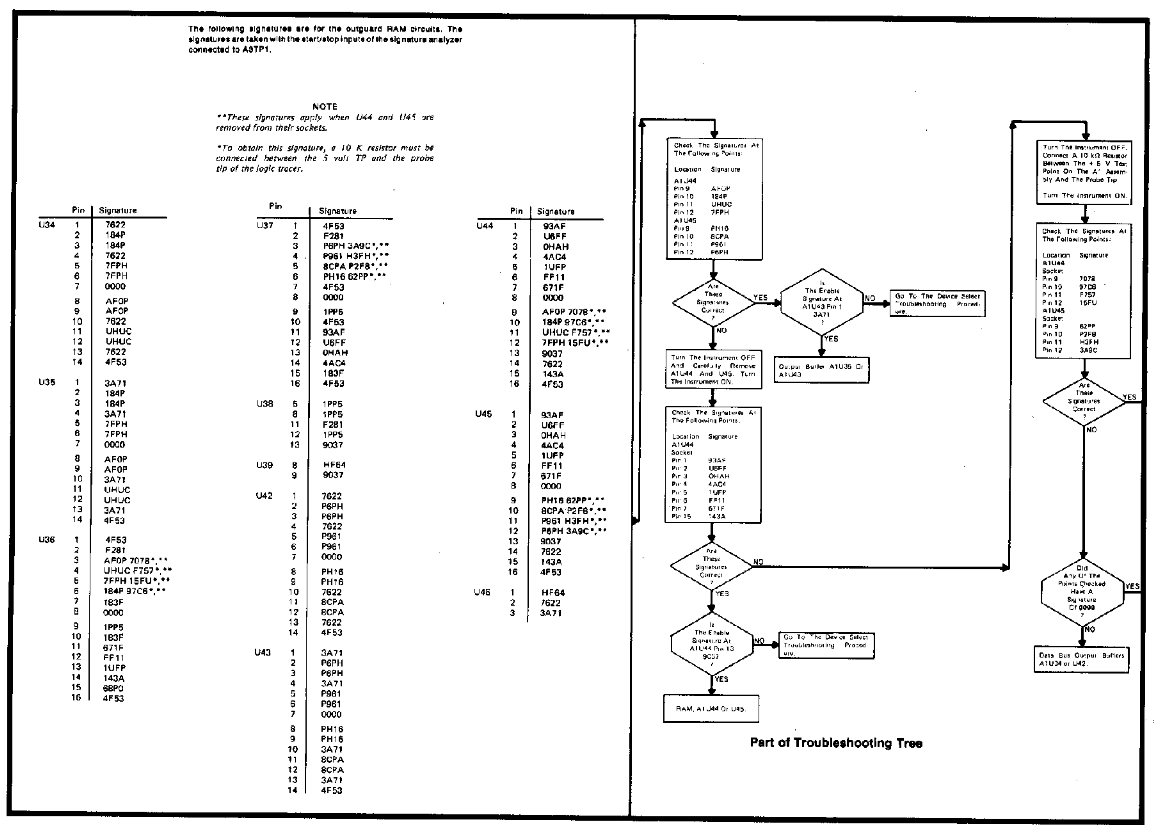


Fig. 8. *An example of a troubleshooting chart from the 3455A Voltmeter service manual. The chart tells which part to replace under certain conditions.*

microprocessor-based instruments relating to troubleshooting and service.

In addition to the authors of the articles in this issue, key contributors include David Kook, who managed to pack the 5004A into an existing plastic case, and Kuni Masuda, who designed the front panel to give it a well-balanced appearance. Gary Gitzen did the breadboarding and designed the unstable signature feature.

It is also a pleasure to acknowledge the many valuable inputs from Dan Kolody and George Haag in Colorado Springs, Kamran Firooz and David Palermo in Loveland, Jan Hofland who is now with the Data Systems Division, Ed White in our own marketing department, and Dick Harris who brought the instrument into production. Gary Gordon as section manager was instrumental in getting the algorithm implemented in a service tool.

Reference

1. A. Gookin, "A Fast-Reading, High-Resolution Voltmeter that Calibrates Itself Automatically," Hewlett-Packard Journal, February 1977.

Hans J. Nadig

Hans Nadig is signature analysis project manager at HP's Santa Clara Division. Holder of an MS degree in electrical engineering from the Federal Institute of Technology in Zürich, he's been with HP since 1967, contributing to the design of instruments for Fourier analysis and serving as a project manager in the IC laboratory. Born in Berne, Switzerland, Hans served two years in the Swiss Army. He's a rock climber, a qualified instructor and guide in high-altitude mountaineering, and a campaigner for environmental protection, especially through power conservation, in local government circles. He also participates in HP's career counseling program for high-school students, and enjoys photography and gardening. Hans is married, has two daughters, and lives in Saratoga, California.

Appendix B

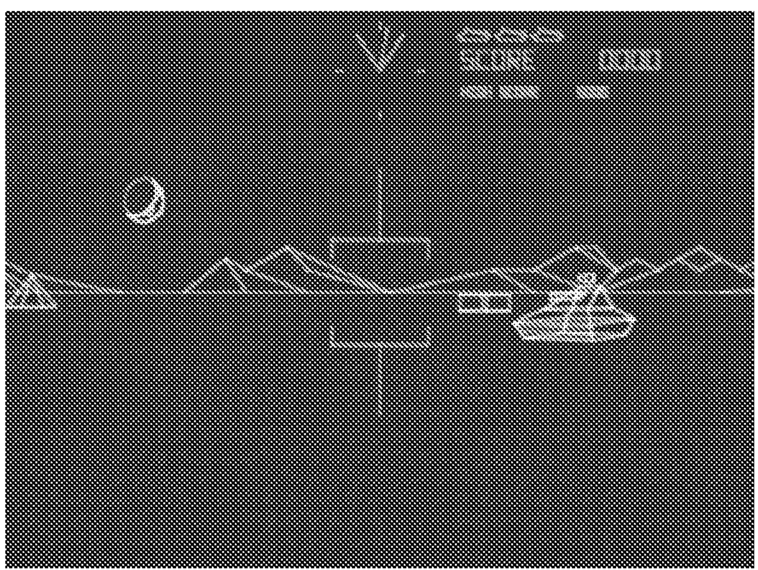
Atari Battlezone ™ Instructions for using Signature Analysis

Background

From: http://www.videotopia.com/games.htm

Battlezone, Atari Inc., 1980. Featuring the first truly interactive 3-D environment, Battlezone so impressed the United States Armed Forces that they commissioned Atari to build specially modified and upgraded versions for use in tank training. The Electronics Conservancy has secured one of these modified games, and it will be displayed in VIDEOTOPIA in the near future.

Screen shot from: **http://tartley.com/wp-content/uploads/2007/05/800px-arcade-atari-battlezone1.png**



From the Schematics

The Auxiliary PCB Math Box Circuitry

The Math Box Circuitry of the Battlezone Auxiliary PCB is connected to the Analog Vector-Generator PCB via the PCB harness interconnector. The Math Box Circuitry receives addresses EAB0 thru EAB4 (external address bus 0 thru 4) and provides data EDB0 thru EDB7 that results in the three-dimensional video of the Battlezone(TM) game.

A second connector on the Auxiliary PCB connects the control signals of the signature analyzer (SA). This header accepts a special harness connector that makes signature analysis extremely easy.

Signature Analysis of the Math Box Circuitry

During the self-test procedure, the Math Box Circuitry is quizzed. **T** displayed in the upper right-hand corner of the self-test video display Indicates that the Math Box Circuitry does not answer the question In the amount of time expected. Therefore, a **T** indicates a Math Box Circuitry failure.

Due to the complexity of this circuitry, we offer signature analysis as a simple means of isolating failing circuits. Signatures for this circuitry are presented In two forms:

1). at the actual test points In the Auxiliary PCB Math Box Circuitry schematic diagram (on Sheet 3. Side B), and

2) for your convenience, on the detail drawing of the Auxiliary PCB to the left of this text.

Since the Analog Vector-Generator PCB must be connected to the Auxiliary PCB, you may take signatures while the PCBs are installed in the game.

The following is the procedure for signature analysis of the Math Box Circuitry of the Auxiliary PCB:

A. Equipment Required:

1. Signature Analyzer (one of the following):

Atari C•A•T Computer-Assisted Troubleshooter. This is a signature analyzer and a RAM/ROM tester combined. For more information contact Atari, Inc., Field Service/Coin-Op Division, P.O. Box 427, Sunnyvale, CA 94086.

OR

Kurz-Kasch Signature II signature analyzer. For more information contact Kurz-Kasch, 711 Hunter Drive, Wilmington, Ohio 45117.

OR

Hewlett-Packard Model 5004A signature analyzer. For more information contact Hewlett-Packard, Scientific Instruments Div., 1501 Page Mill Road, Palo Alto, CA 94304.

For local dealers, check the Yellow Pages under "Electronic Equipment and Supplies."'

2. SA Harness Assembly:

   Atari part number A036836-01. You can make one of these yourself. Above is an illustration of its construction.

3. Three jumper wires with "hook" connectors on each end.

4. Pullup resistor as follows: 1K to 1.5K ohm, 1/4 watt resistor.

B. Signature Analysis Setup Procedure

1. Connect Signature Analyzer to the matching pins of SA connector on the SA Harness assembly. In other words, GND should match up with GND, etc.

2. Set Self-Test Switch of Battlezone(TM) game to ON. After approximately three seconds, the TV monitor should display the self-test pattern.

3. Jumper top end of 1K-ohm resistor R129 (located immediately between and below C [center] and L [left) COIN test points of Analog Vector-Generator PCB to ground five times, or until video display Is blank. You will hear a short beep after the 5th grounding, also, the screen will display only a tiny dot In its center. **NOTE:** To avoid accidentally turning off the game by brushing against the interlock switch, we recommend putting tape over the switch.

Alternate: Jumper pin 5 of Analog Vector-Generator PCB edge-connector J20 to ground five times, or until video display is blank.

C. Signature Analysis Test #1 Procedure

1. Plug SA Harness Assembly Test #1 connector onto Signal Analyzer header on Auxiliary PCB (the black wire on the connector should be at the top.

2. Connect a jumper between pin 1 of IC B6 on the Analog Vector-Generator PCB and ground. This places a continuous RESET to the microprocessor on the Analog Vector-Generator PCB.

3. Set Signature Analyzer START to ,STOP to and CLOCK to .

4. Connect a jumper wire to each and of a 1K to 1.5K-ohm resistor. Connect one jumper wire to +5V test point on Auxiliary PCB. Connect other jumper wire to the tip of the Signature Analyzer probe.

5. Verify that setup procedure was correct by probing (touching probe to) the +5V test point. The Signature Analyzer should indicate **CC34**. If not **CC34**, remove the jumper from pin 1 of IC B6. Return to *B. Signature Analysis Setup Procedure* and once again do step 3.

6. Probe for signatures as shown In Figure 1 to the left. It all signatures are correct, continue with *D. Signature Analysis Test #2A Procedure.* If any signatures are incorrect, probe for signature at **CC34** on +5V test point. If not **CC34,** remove jumper from pin 1 of IC B6. Return to *B. Signature Analysis Setup Procedure* and once again do step 3. If +5V is **CC34**, refer to *G. Isolating a Failing Circuit,*

D. Signature Analysis Test #2A Procedure

1. Remove 1K to 1.5K-ohm jumper wire from Signature Analyzer probe.

2. Plug SA Harness Assembly Test #2 connector onto Signature Analyzer header on Auxiliary PCB.

3. Remove jumper from pin 1 of IC B6 on the Analog Vector-Generator PCB.

4. Set Signature Analyzer START to ,STOP to and CLOCK to .

5. Verify that setup procedure was correct by probing + 5V for a signature of **3951**. If not **3951**, return to *B. Signature Analysis Setup Procedure* and once again do step 3, then return to this step.

6. Probe for signatures as shown in Figure #2A to the left. If all signatures are correct, continue with *E. Signature Analysis Test #2B Procedure.* If a signature is incorrect, refer to *Isolating a Failing Circuit.*

E. Signature Analysis Test #2B Procedure

1. Make sure the SA Harness Assembly Test #2 connector is plugged onto Signature Analyzer header on Auxiliary PCB.

2. Make sure jumper is removed from pin 1 of IC B6 on the Analog Vector-Generator PCB.

3. Set Signature Analyzer START to _/‾ ,STOP to ‾_ and CLOCK to _/‾ .

4. Verify that setup procedure was correct by probing +5V for a signature of **3951**, if not **3951**, return to *B. Signature Analysis Setup Procedure* and once again do step 3, then return to this step.

5. Probe for signatures as shown In figure #2B to the left. If all signatures are correct, continue with *F. Signature Analysis Test #3 Procedure.* If a signature is incorrect, refer to *G. Isolating a Failing Circuit.*

F. Signature Analysis Test #3 Procedure

1. Plug SA Harness Assembly Test *#3* connector onto Signature Analyzer header on Auxiliary PCB.

2. Make sure jumper is removed from pin 1 of IC B6 on the Analog Vector-Generator PCB.

3. Set Signature Analyzer START to _/‾ ,STOP to ‾_ and CLOCK to _/‾ .

4. Verify that setup procedure was correct by probing +5V for **3951**. If not **3951**, return to *B. Signature Analysis Setup Procedure* and once again do step 3, then return to this step.

5. Probe for signatures as shown in Figure #3 to the left. If all signatures are correct, then Math Box Circuitry of Analog Vector-Generator PCB is OK.

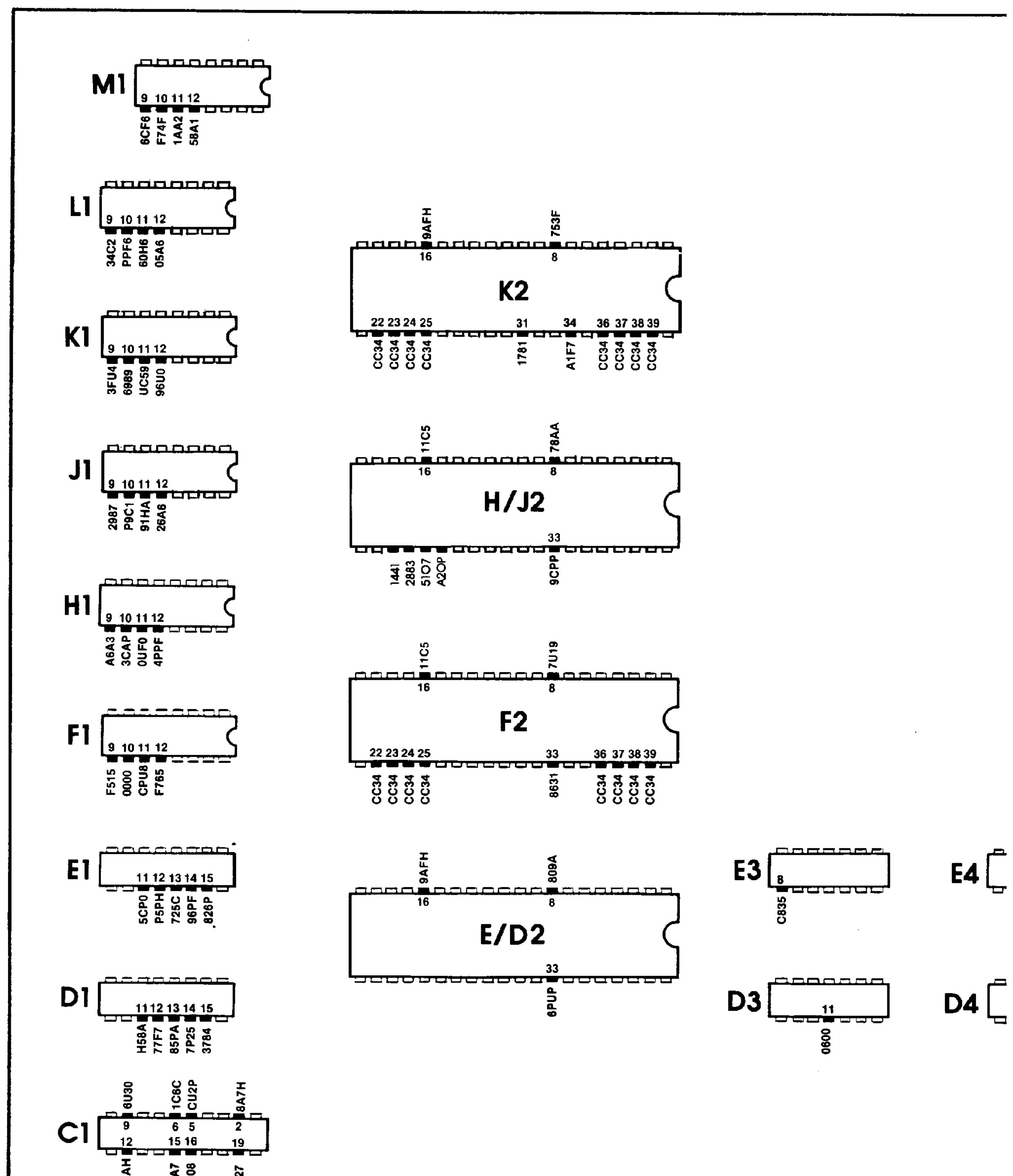
G. Isolating a Failing Circuit

If you find an incorrect signature, find the signature test point of the Math Box Circuitry on Sheet 3, Side B. Locate the IC from which the signature is being output. Check all inputs of that IC.

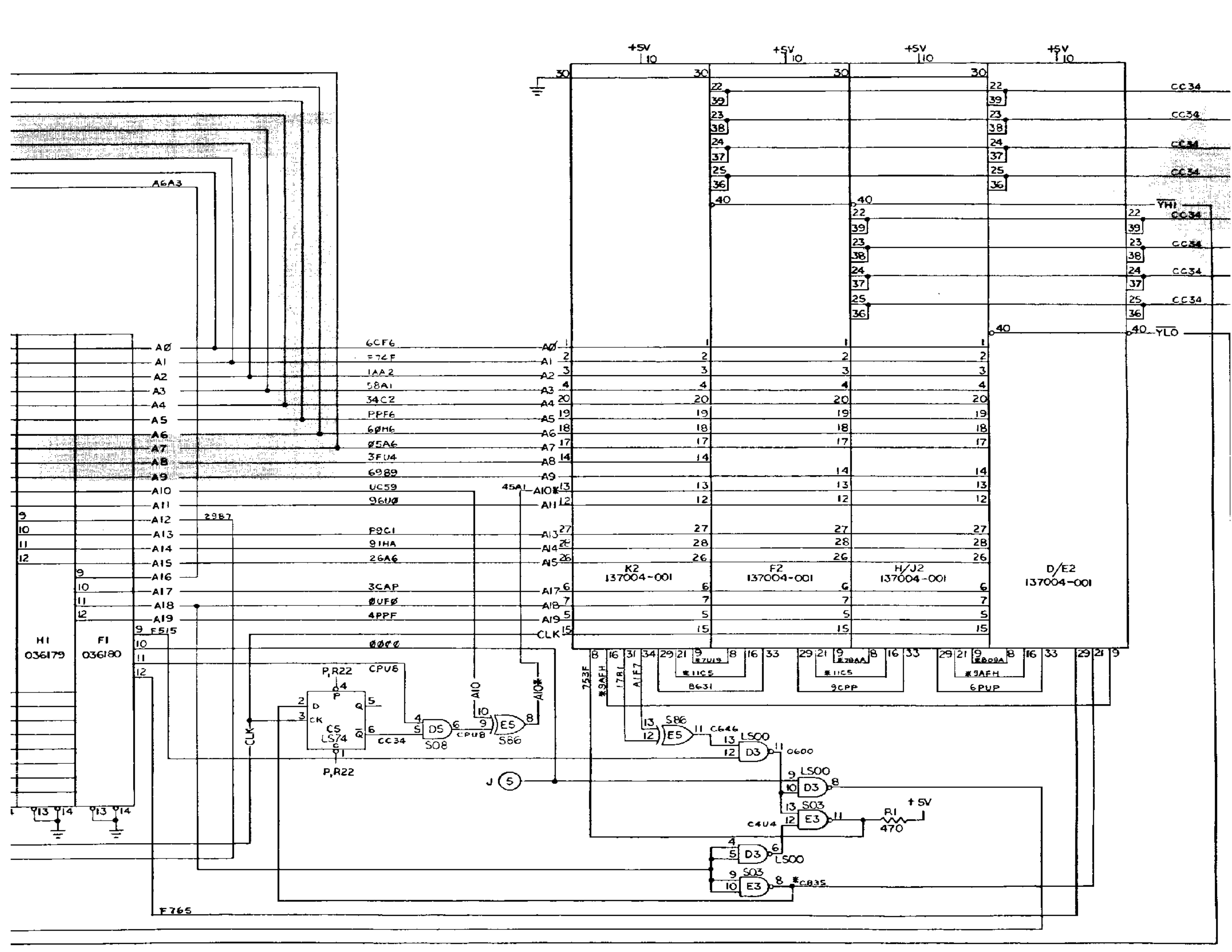
If all Input signatures are correct: Remove the Auxiliary PCB from the circuit. Check the circuit traces common to the failing IC pin on both the top and bottom of the PCB for shorts to another circuit trace. If the circuit traces are not shorted, then replace the failing IC.

If an Input signature Is incorrect: Locate on the schematic the IC source of the failing signature. Check the input signatures of that IC. If all input signatures are correct, then that is the failing IC. If this IC has a failing input signature, then continue "upstream" in the circuit flow until the failing IC is isolated.

A Section from the Battlezone schematics showing a pictorial representation of pin locations and signatures.



A section of the schematics showing signatures.



* **All signatures with an asterisk are taken with a 1K ohm pull-up resistor attached between the signature analyzer data probe and +5 VDC.**

The full set of Battlezone schematics can be downloaded at
www.jmargolin.com/bz/BZ_DP-156 2nd Printing.pdf (2.3 Mbytes PDF)

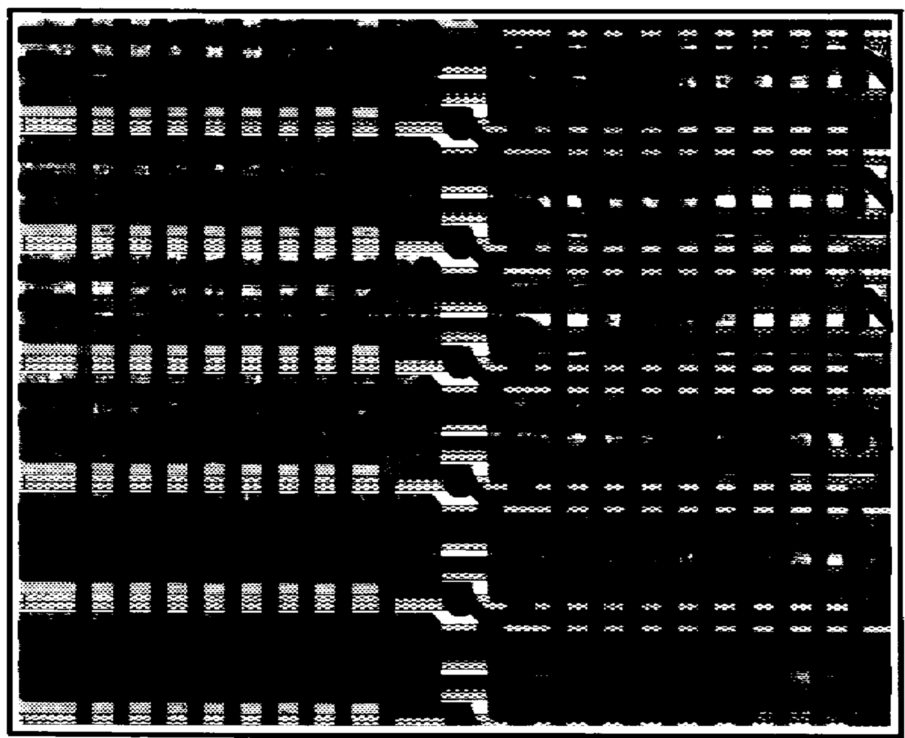
Jed Margolin
Reno, NV
September 10, 2007

Appendix C

SEMICONDUCTOR MEMORIES

A Handbook of Design, Manufacture, and Application

Second Edition



Betty Prince

Semiconductor Memories

A Handbook of Design, Manufacture and Application

Second Edition

Betty Prince
Texas Instruments, USA

JOHN WILEY & SONS
Chichester • New York • Brisbane • Toronto • Singapore

Second edition of the book *Semiconductor Memories* by B. Prince and G. Due-Gundersen

Copyright © 1983, 1991 by John Wiley & Sons Ltd.
Baffins Lane, Chichester
West Sussex PO19 1UD, England

All rights reserved.

No part of this book may be reproduced by any means,
or transmitted, or translated into a machine language
without the written permission of the publisher.

Other Wiley Editorial Offices

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158-0012, USA

Jacaranda Wiley Ltd, G.P.O. Box 859, Brisbane,
Queensland 4001, Australia

John Wiley & Sons (Canada) Ltd, 22 Worcester Road,
Rexdale, Ontario M9W 1L1, Canada

John Wiley & Sons (SEA) Pte Ltd, 37 Jalan Pemimpin #05-04,
Block B, Union Industrial Building, Singapore 2057

Library of Congress Cataloging-in-Publication Data

Prince, Betty.
Semiconductor memories / Betty Prince. — 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0 471 92465 2
1. Semiconductor storage devices. I. Title.
TK7895.M4P74 1991 91-6943
621.39'732—dc20 CIP

British Library Cataloguing-in-Publication Data

Prince, Betty
Semiconductor memories. — 2nd ed.
I. Title
621.3815

ISBN 0 471 92465 2

Typeset by Techset Composition Limited, Salisbury, Wiltshire
Printed in Great Britain by Courier International, East Kilbride

ACKNOWLEDGEMENTS

I would like to thank all those who contributed information and the many experts in various memory fields who proof read chapters and offered suggestions. In particular Roelof Salters, who is technical advisor to the Philips Advanced Memory Design Center and Design Manager for the JESSI Design Team at Philips, for advice and suggestions throughout the book.

Fred Jones of Dataquest, Con Gordon and Pieter te Booij of Philips, and Jim Benson of TI for reading and offering helpful suggestions on Chapters 1 and 2.

Hein van Bruck and Rien Galema of the Philips Central Applications Laboratory, and Bill Vogley of TI for reviewing and for helpful suggestions on Chapter 3.

Ad Bermans, of the European JESSI Office, formerly Advanced Technology Manager of the Philips I.C. Technology center, and Maartin Vertregt of the Philips Advanced Memory Design Center for reviewing and for suggestions on Chapter 4.

Cormack O'Connell, Design Team Leader at Mosaid, and Howard Sussman, Design Manager at NEC, for reviewing and for many suggestions on Chapter 5.

Howard Sussman, and Ad van Zanten, Design Manager of the Philips Advanced Memory Design Centre, for suggestions on Chapters 6 and 7 and Joe Hartigen of TI for comments on Chapter 7.

Frans List, SRAM Design Managers at Inmos, and Ad van Zanten for reviewing and for many helpful suggestions on Chapters 8 and 9.

Roger Cuppens, Non-volatile Design Manager of the Philips Advanced Memory Design Center, for reviewing and offering suggestions on Chapters 10, 11 and 12. Sebastiano D'Arrigo, flash EPROM Design Manager, of T.I. for reviewing and offering helpful suggestions on Chapter 11, Don Knowlton of Waferscale Integration for reviewing Chapters 11 and 12 and Howard Sussman of NEC for reviewing Chapter 10.

Henk Kiela of the Philips Package Development Group and Daniel Baudouin of TI for suggestions on Chapter 13.

Reese Brown, who is now a private consultant and formerly Components Manager of Unisys, for helpful suggestions on the reliability aspects of Chapter 14, and J. Weidenhofer of the DRAM Test Department at Siemens and John Salter, Production Engineering Manager at the Philips Submicron Technology Facility, for help on the test sections of this chapter.

Albert Maringer of Siemens for contributing material and for helpful suggestions on Chapter 15. Frank Stein and Sharon Phelan for helpful suggestions on editing.

Pallab Chatterjee of TI whose inspiring opening talk of the 1990 VLSI Technology

Symposium contributed to the writing of Chapter 16, and Earnest Powell of TI for suggestions on Chapter 16.

Finally I would like to thank Dr Theo Claasen, Director of Design at Philips, for his inspiration and encouragement throughout this book.

While the author gratefully acknowledges the above noted comments and suggestions, all errors in the final version remain strictly the responsibility of the author, who would greatly appreciate being notified of such errors so they can be corrected in subsequent printings.

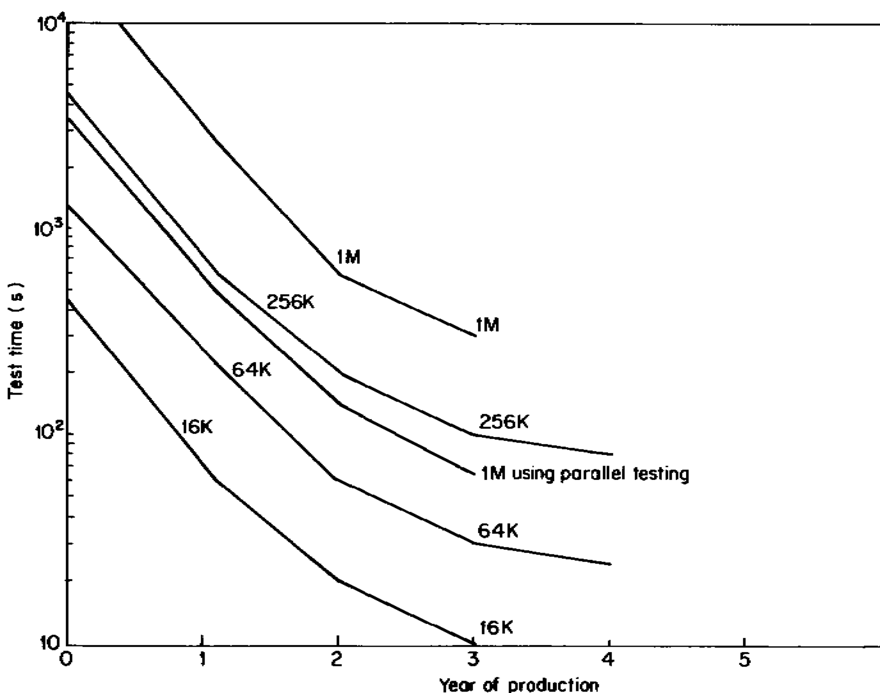


Figure 14.1
Final test time for various DRAM densities by year of production (assuming similar test flow).

Highly sophisticated test algorithms have been developed with the aim of reducing test time while maintaining the required quality.

Figure 14.1 illustrates test timing improvements for various densities of DRAMs from the point of device introduction followed by an engineering experience curve until an optimum and stable situation is reached in about the third year of production.

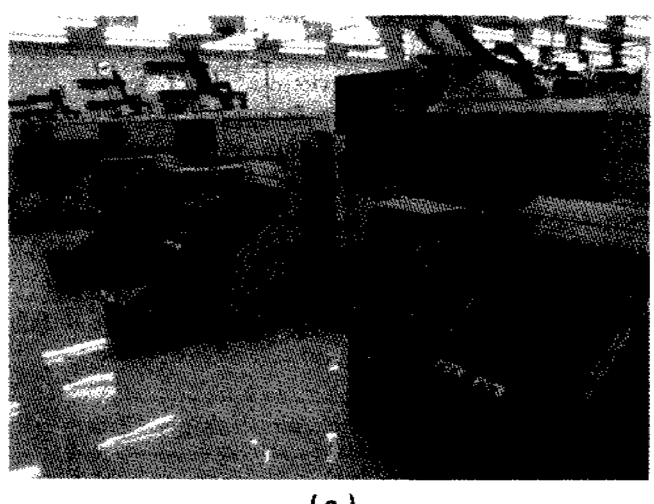
The average test time required for the different types of MOS memory products varies considerably. DRAMs and SRAMs of similar densities have similar test times. Products such as EPROMs take a factor of 10 increase in test time. Since test time impacts the throughput of the product line and is allocated in overhead on expensive test equipment it adds considerably to the cost of production. Figures 14.2(a) and (b) show memory test equipment being used.

14.2 FAILURE MODES AND TEST PATTERNS

The rapid growth in circuit complexity has greatly increased the difficulty and cost of testing memories.

Semiconductor memory testing can be basically reduced to four different groups:

Cell test Every cell must be capable of storing a logic '0' and a logic '1' for a given minimum amount of time and, if the memory is writable, must be capable of being changed from one state to the other.



(a)

(b)

Figure 14.2
Examples of memory test. (a) Volume production final electrical test showing memory test machines and handlers. (b) Sample electrical test at outgoing inspection. ([60] 1990, with permission of NMB Semiconductor.)

Data in–out test Sense lines and data in–out lines must be capable of recovering from read–write operations. Sense amplifiers have to operate within the small, specified voltage or charge levels.

Address decoder test Every cell must be correctly and uniquely addressed by the decode logic.

Disturb testing The accessing of one part of the memory array must not affect any other part.

Special test Functions specified for different memory types must be checked and verified operational. This will include the testing of control lines such as chip enable, output enable, chip select, or special logic circuits.

The design of a sequence of test patterns to check the necessary failure condition for a memory device calls for not only thorough component knowledge, but also an in-depth understanding of various MOS failure mechanisms.

It is important to test for the maximum number of failure modes for the test pattern chosen, while at the same time avoiding combinations of standard patterns which result in double testing, or in checking for unnecessary conditions.

A few common memory failure modes highlighting the above failure groups are listed.

- Cell: 'stuck-at' failures, open or short circuits, leakage, adjacent cell disturbance.
- Access times: minimum and maximum to specification.
- Address decoder: open or short circuits, noise, high sensitivity.
- Sense amplifier: recovery time.
- Refresh times: minimum.
- Clocks.
- Write: recovery time.

'Stuck-at' cell failures are one of the commonest forms of failure. In this mode the single cell bit is simply 'stuck' at '1' or '0'. Complex tests are not necessary to determine this type of failure.

Additional failure modes specifically related to different memory families also have to be screened for and will add to the above list.

A variety of standard test patterns are commonly used for screening out most known failures. Figure 14.3(a) illustrates some test pattern types with corresponding test time constants and failure descriptions.

The test time required is calculated by substituting N with memory size (or number of bits) followed by multiplication by the cycle time used. It can be seen that most tests fall within three different categories: $2N$, $2N^{3/2}$, or N^2. Test time is mostly dependent on memory size, cycle time, and test pattern performed.

Figure 14.3(b) illustrates test times as a function of memory size for $2N$, $2N^{3/2}$, and $2N^2$ type patterns.

A typical test pattern sequence might include the following.

1. All '1' or all '0'.
2. Checkerboard

Pattern	Constant	Failure
March	$10N$	Cell access
Walking pattern	$2(N^2 + N)$	Multiple address sense amplifier recovery
Galloping pattern	$2(N^2 + N)$	Multiple address sense amplifier recovery
Sliding diagonal	$2(3N^{3/2} + 5N)$	Cell testing, diagonal
Galloping column	$2(3N^{3/2} + 6N)$	Cell testing, columns

(a)

Pattern	1k	4k	16k	64k	256k
$2N$	0.41	1.64	6.55	26.2	76.8
$2N^{3/2}$	13.57	104.8	839	6710.9	53,687
$2N^2$	439	6710.9	107 347	1717 987	27 487 792

(b)

Figure 14.3
(a) Typical test patterns with corresponding test time constants and failure descriptions 'N' is the device bit density. (b) Illustrative test times as a function of density. (Reproduced with permission of John Wiley.)

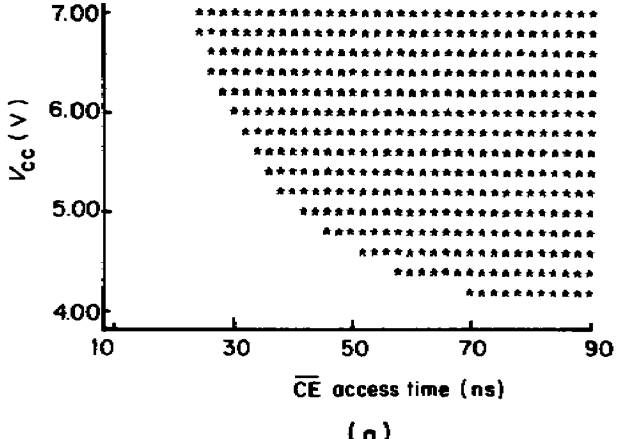
3. Stripe
4. Marching
5. Galloping
6. Sliding diagonal
7. Waling
8. Ping-Pong.

Numbers 1 to 4 are called '*N*' patterns. These can check one sequence of N bits of memory by at most using the given pattern several times. Numbers 5 to 7 are called N^2 patterns. These need several times of N^2 patterns to check one sequence of N bits of memory. N^2 patterns have a long test time for high density memories. For example, a 64k RAM takes about 30 minutes to test with a galloping pattern and a 1Mb RAM takes over 6 hours.

The first three patterns in the sequence shown can check the array but are not sufficient to check the decoder circuits. The marching pattern is the simplest pattern which will check out the function of the memory. A description follows as an example of a typical marching diagonal test pattern.

A marching pattern is a pattern in which '1's march into all '0's. The procedure is as follows.

1. Clear all bits to '0'.
2. Read '0' from '0'th address and check that read data are '0'.
3. Write '1' on 0th address.
4. Read '0' from 1st address, and check read data are '0'.



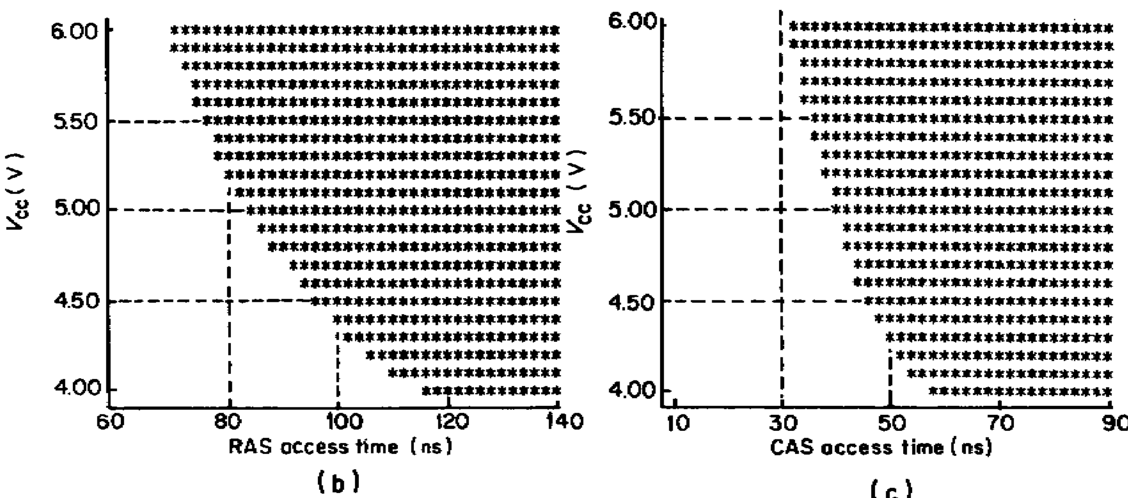


Figure 14.4
Various Shmoo plots showing functional device parameter regions for DRAMs. (a) V_{CC} plotted against $\overline{CE}$ (From Benevit *et al.* [37], AT & T 1982, with permission of IEEE), (b) V_{CC} plotted against $\overline{RAS}$, (c) V_{CC} plotted against $\overline{CAS}$. (From Kantz [45], Siemens 1984, with permission of IEEE.)

5. Write '1' on 1st address.
6. Repeat for N addresses until all are '1's.
7. Repeat 2 to 6 reading '1's and writing '0's until all data are '0'.

The commonest kind of 'stuck-at' failures can be found by this type of test. Test information can be presented as follows.

- Shmoo plots show the interaction between device parameters by plotting the range of functional parameter sets. Figure 14.4(a) shows the relation between V_{CC} and $\overline{CE}$ access time on a 256k DRAM. The plot shows a 40 ns chip enable access time at 5 V for the device under test. Figure 14.4(b) shows V_{CC} as a function of $\overline{RAS}$ access time and Figure 14.4(c) shows V_{CC} as a function of $\overline{CAS}$ access time.

  Asterisks denote the regions where the part is functional. Shmoos are generally used for device characterization, intersystem correlation, test program development, process and device correlation, and failure analysis.

- Accumulative Shmoo plots are derived from superimposing several Shmoos from different components.

- Wafer mapping is shown in Figure 14.5(a). This is used for probe yield analysis, wafer fabrication defects analysis, mask defect analysis, alignment problems, and system correlation. This is an accumulative representation of several wafers where the total failures for each die location are given as a percentage.

- Bit mapping: this gives fail–pass information for every bit or cell location of the memory under test. The results can be displayed on a color graphics terminal or printed. Real time bit mapping is fast and very effective during failure analysis, process evaluation, probe yield enhancement, pattern sensitivity detection, data retention (EPROMs) testing, pattern verification (ROMs) and device characterization. Figure 14.5(b) for example, shows a bit map of a 16k RAM as a 178 by 178 matrix. The vertical dotted line illustrates a number of faulty bits.

14.3 TESTING DRAMs AND SRAMs

The various memory types of families such as ROMs, EPROMs, or RAMs will have different testing requirements due to differences in device characteristics, failure mechanisms, and operating modes. We will discuss special test considerations for dynamic and static RAMs, EPROMs, EEPROMs, and embedded memories.

14.3.1 Fault coverage considerations

The major problem in RAM testing is obtaining good fault coverage. As RAM density increases and advanced technologies and circuit techniques bring with them more complex failure modes, testing time increases rapidly.

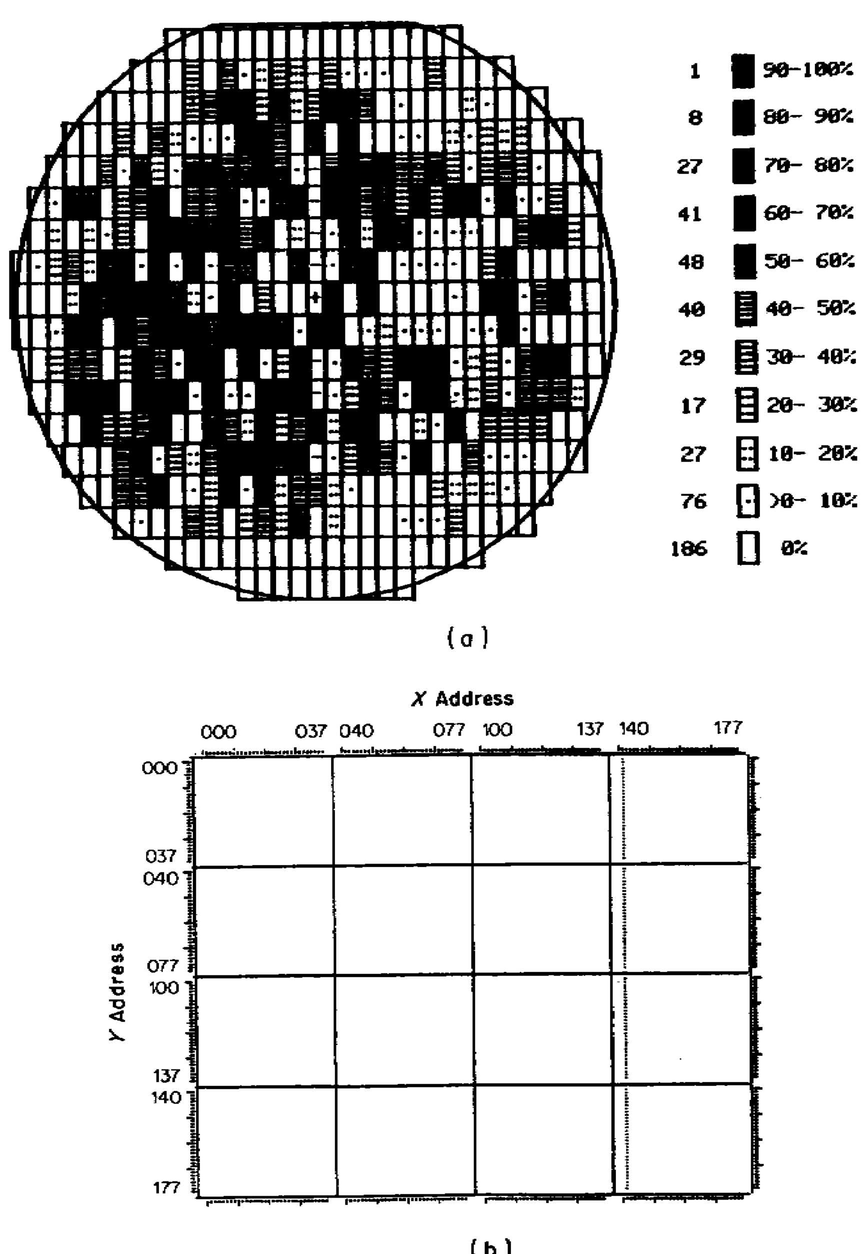


Figure 14.5
(a) Wafer mapping of 16k DRAM. (b) Bit map of 16k DRAM. (From Prince and Due-Gundersen [64] 1983.)

For example a simple sliding diagonal test requires several hours to perform on a single 1Mb RAM chip. The more complex galloping pattern (GALPAT) test requires a testing time proportional to N^2 for an N-bit RAM. If testing a 1k RAM takes a few seconds, then testing a 1Mb RAM may require several days. The extent of fault coverage in these tests is also not easy to define.

Table 14.1 Normalized test times for dynamic RAMs.

Density	Normalized test time
65k	1.0
256k	3.1
1M	10.4

Normalized test times for various generations of DRAM are shown in Table 14.1 taken from a paper by TI [50] where the test time of the 64k DRAM is taken as '1' and it is assumed that the same test flow is used for all. If 8 bit parallel testing is used for the 1Mb DRAM then the test time is reduced from a factor of 10.4 to 2.0. Parallel testing is described more fully in the later section on DRAM test modes.

14.3.2 Failure modes

Most failure modes related to RAMs are well known with corresponding test patterns for effective testing. Typical RAM failure modes include

- 'stuck at' faults
- pattern sensitivity
- multiple writing
- refresh sensitivity (DRAMs)
- open–short circuits
- leakage current faults
- sense amplifier recovery
- access time
- voltage bump (DRAMs)

A combination of well known $N^{3/2}$ and N type test patterns will generally provide the solution to effective screening for most of these failure modes. When the characteristics of a specific device are known it is often possible to delete parts of the initial test program and obtain a substantial reduction of test times.

Mechanisms, which are frequently involved in these failure modes, are reviewed below.

Gate oxide defects can cause stuck-at faults such as stuck-at 0, stuck-at word line, bit-word line crosstalk, transmission line effects, and row decoder failures.

The major sources of pattern sensitive failure modes in DRAMs are neighborhood interference faults, sense amplifier recovery problems and bit-line imbalance faults.

Neighborhood interference faults are frequently due to leakage current mechanisms

which can depend on the pattern of stored data in the neighboring cells. When the leakage current flowing between one cell and another is sufficient to destroy the contents of the cell, a neighborhood interference fault is said to occur. The worst case is when all surrounding cells have the opposite state to the tested cell.

Sense amplifier recovery problems can be caused by parasitic capacitance and resistance, which can also cause slow sense amplifiers, and transmission line effects. Sense amplifier recovery faults occur when, after repeated writing of the same cell, the data read out from that cell are independent of the contents of the cell.

Bit-line imbalance faults are caused by the difference in the total leakage associated with the cells connected to the two bit-lines involved.

14.3.3 Voltage bump test for DRAMs

Voltage bumping, or fluctuations of the power supply voltage, can cause erroneous data to be read out of the RAM. The voltage bump problem demands some special testing and the voltage bump test is a particularly rigorous test for a DRAM.

A positive V-bump is defined as V_{CC} during the write operation being lower than V_{CC} during the read operation. This fluctuation of V_{CC} lowers the read out voltage from the memory cell and may cause a read error.

If a V_{CC} level cell plate is used, the positive V-bump raises the stored level in the cell. Since the dummy cell level is kept at V_{SS} in spite of the bump, a low level in the cell may read erroneously as a high level.

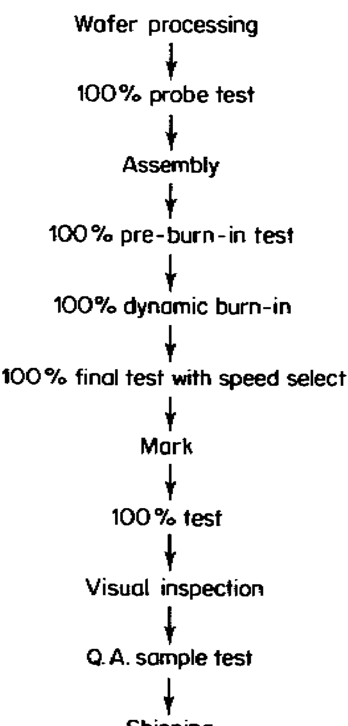


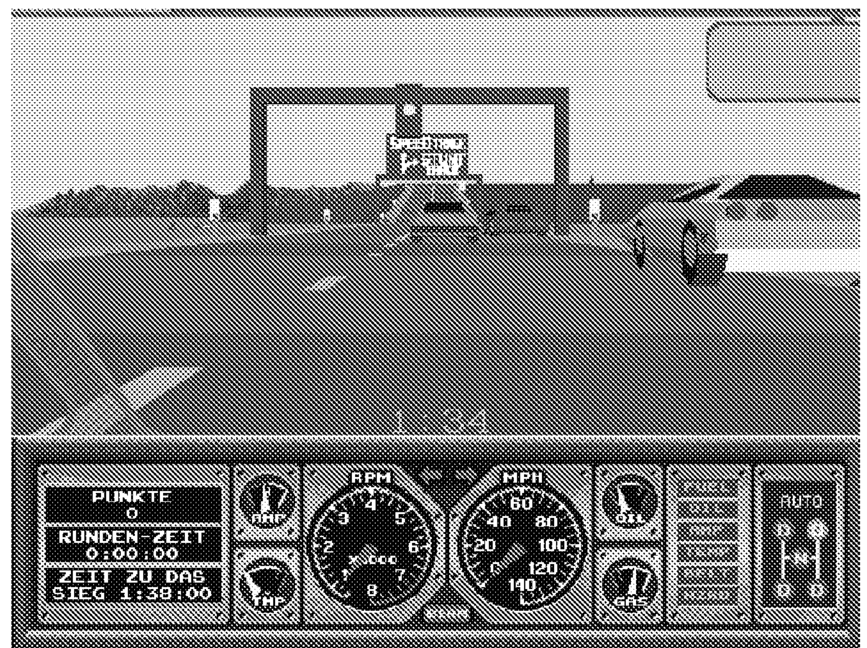
Figure 14.6
Typical dynamic RAM production flow charts. (From Prince and Due-Gundersen [64] 1983.)

Appendix D

From: **http://www.arcade-history.com/?n=hard-drivin'&page=detail&id=1083**

arcadehistory

Earth's biggest coin-operated machine database ...featuring **19051** machines



[Video Game] Hard Drivin' © Atari Games (1988)

DESCRIPTION

GENRES : **Driving**
Type of the machine : **Video Game**

Hard Drivin' © 1988 Atari Games.

Slide into the contoured seat and adjust it to fit the length of your arms and legs. Put your feet on gas and clutch pedals and try the stick shift. Select manual or automatic transmission, turn the ignition key and you're off!

It's the ride of your life. You feel the tires grip the road when you take a wide turn at high speed. You're alerted to the smallest change in the road by the feedback steering. You catch air as you fly the draw bridge and land on the down ramp. You control the car as it holds the road on the dizzying vertical loop.

Hard Drivin' might look like an arcade game but it drives like a real car. For the best lap times, drive Hard Drivin' as if it were a real car. The main difference between Hard Drivin' and a real car is that Hard Drivin' is much safer to drive. A player can test the limits of our car and his skill with no risk of personal injury, and follow a course that does not exist anywhere m the real world.

After inserting the proper number of coins to start the simulator, the player can select either an automatic or manual transmission. Turning the ignition key starts the simulator.

Drivers can choose between the stunt track or the speed track by following the posted signs on the road. Each player has a certain (operator-selectable) amount of time to reach a checkpoint or the finish line. Crossing checkpoints and the finish line rewards the player with (operator-selectable) bonus driving time.

With Hard Drivin' a player can test drive a high-powered sports car on a real stunt course. He can jump a draw bridge, negotiate a high-speed banked turn and drive a 360-degree vertical loop. These thrilling stunts, among others, provide the ultimate realistic driving experience.

Or maybe high-speed driving is a particular player's type of excitement. He can 'put the pedal to the metal' and try to keep control around the corners, weaving in and out of traffic while avoiding oncoming cars. All this, and more, await the player behind the wheel of Hard Drivin'.

Players especially enjoy the unique instant replay feature on Hard Drivin'. Every time a player crashes, the simulator records and replays the crash sequence. Not only will the player find this entertaining, but it is also informative. The instant replay shows the player exactly what he did wrong and why he crashed (If a player wants to skip the instant replay, he can press the abort switch or turn the key when the replay starts).

A skilled player finds the ultimate competition in the 'challenge' lap (or 'grudge match' as Atari Games likes to call it). The simulator remembers the path of the car driven by the best driver on record. When a player beats the qualifying lap time, he challenges the car of the past winner in a head-to-head race.

TECHNICAL

Game ID : 136052 (cockpit), 136068 (compact)

Main CPU : 68010 (@ 8 Mhz), TMS34010 (@ 6 Mhz), TMS34010 (@ 6.25 Mhz), ADSP2100 (@ 8 Mhz)
Sound CPU : 68000 (@ 8 Mhz), TMS32010 (@ 5 Mhz)
Sound Chips : DAC

Screen orientation : Horizontal
Video resolution : 508 x 384 pixels
Screen refresh : 60.00 Hz
Palette colors : 1024

Hard Drivin' is equipped with center-feel steering with continuous force feedback, adjustable swivel seat, gas, brake and clutch pedals, four-speed stick shift, and a medium-resolution monitor.

TRIVIA

This was the world first driving simulator to use 3-D polygon graphics.

Despite claiming to be a real driving simulator, there were a lot of discrepancies between the game's software physics and the car physics on screen. However, the cockpit physics were considered very accurate at the time.

You may have noticed that the Credit Screen lists Doug Milliken as a Test Driver (See Staff section). He is listed as a Test Driver because Atari didn't want anyone to know what he really did. Hard Drivin' had to be as accurate as possible. That meant doing an accurate car model to mathematically describe the physics of how the parts of the car (engine, transmission, springs, shock absorbers, tires, etc.) react to each other, to the road and to the driver's inputs. The pioneer in the field (in the 1950s) was William Milliken of Milliken Research. His son, Doug, has continued his father's work. Doug is probably the world's leading expert in car modeling. Doug and his father wrote the book on car modeling.

Patents that come out of Hard Drivin' are :

5,005,148 : 'Driving simulator with moving painted dashboard'.
5,354,202 : 'System and method for driver training'.
5,577,913 : 'System and method for driver training with multiple driver competition'.

Prior to the release of Hard Drivin', Namco had acquired a controlling interest in Atari games by 1986. The sharing of R&D information would spawn many games of the same polygon engine years later. It can be credited that the success of the Hard Drivin' engine set the trend for the high quality simulation games in the early 90's.

One of the buildings along the speed course, a small camouflage-painted building, if approached from behind (a non-trivial task, given the off-road time limit) has a sign above its normally-unseen door that says 'THE HUT'.

If the driver slowed down and stopped in front of one of the buildings, a 'keyhole' appeared on the building's door.

There is no apparent Ferrari license shown in any version of the game.

Jerry Landers holds the official record for this game with 529,800 points.

There were 15 officially released versions, counting 11 cockpit and 4 compact versions, including various British, German and Japanese versions.

A free, playable version of Hard Drivin' was displayed in the Franklin Institute Science Museum in Philadelphia, Pennsylvania in 1989.

UPDATES

Notes : In all British versions, you are in a right-hand drive car.

*** Cockpit versions :**

Revision 1 :
* World release.
* Software version : 7.8.

Revision 2 :
* World release.
* Software version : 7.9.

Revision 3 :
* World release.
* Software version : 8.1.

Revision 4 :
* German release only.
* Software version : 8.2.

Revision 5 :
* British release only.
* Software version : 8.3.

Revision 6
* British and Japanese releases only.
* Software version : 8.4 for Japanese and 8.5 for British.

Revision 7
* World, British and Japanese release.
* Software version : 8.6 for all.

*** Compact versions :**

Revision 1 :
* World release.
* Software version : 2.9.

Revision 2 :
* World, British and German releases.
* Software version : 3.1 for all.

TIPS AND TRICKS

If the driver made a hard left turn at the start of the game, a 'secret' track was available. The track was a long straight road leading to a very short circular track (a skid pad test track) around a tower.
SERIES

1. Hard Drivin' (1988)
2. Race Drivin' (1990)
3. Hard Drivin' II - Drive Harder (1991, Atari ST, Commodore Amiga)
4. Hard Drivin's Airborne (1993)
5. Street Drivin' (1993)

STAFF

*** Main :**

Project leader, game designer, sound system, mech designer, force shifter, analog HW: **Rick Moncrief**

Techician, mech, designer, sound recording, dashboard shift, game designer: **Erik Durfey**

Software designer, game designer, car model, force feedback steering, SW tools: **Max Behensky**

Hardware designer, self test, instant replay, integer 3D algorithms, game designer: **Jed Margolin**

Game programming, display software, championship lap, game designer: **Stephanie Mott**

*** Others :**

Cabinet designers : **Mike Jang, Ken Hata**

Graphics : **Sam Comstock, Kris Moser, Deborah Short, Alan Murphy**

Display math software : **Jim Morris**

ADDN'L programming : **Gary Stark, Mike Albaugh, Ed Rotberg**

ADDN'L hardware : **Don Paauw**

Marketing : **Linda Benzler, Mary Fujihara**

Sales : **Shane Breaks**

Mechanical designers : **Jacques Acknin, Milt Loper, Geoff Barker**

Test drivers : **Doug Milliken, Dave Shepperd**

Music : **Don Diekneite**

Management : **Dan Van Elderen, Lyle Rains, Hide Nakajima**

PORTS

Consoles :

Sega Mega Drive (1990)
Atari Lynx (1991)
Microsoft XBOX (2004, "Midway Arcade Treasures 2")
Nintendo GameCube (2004, "Midway Arcade Treasures 2")
Sony PlayStation 2 (2004, "Midway Arcade Treasures 2")

Computers :

Commodore C64 (1989)
Commodore Amiga (1989)
Atari ST (1989)
Amstrad CPC (1989)
Sinclair ZX Spectrum (1990)
PC [MS-DOS] (1990)
PC [MS Windows, CD-ROM] (2006, "Midway Arcade Treasures Deluxe Edition")

Notes : Upon purchasing the Amiga version, a questionnaire contest was held where the first 5 people to answer correctly via a postcard sent to London would receive a free model Ferrari F-40 model car by January 8, 1990. The model car is 1/18th the size of the actual car.

LAST EDITION
August 17, 2007