NASA Contractor Report 166073

NASA-CR-166073
19850022395

DEVELOPMENT AND EVALUATION OF A FAULT-TOLERANT MULTIPROCESSOR (FTMP) COMPUTER
Volume III
FTMP Test and Evaluation

Jaynarayan H. Lala and T. Basil Smith, III

THE CHARLES STARK DRAPER LABORATORY, INC.
555 Technology Square
Cambridge, Massachusetts 02139

CONTRACT NAS1-15336
MAY 1983

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

FOREWORD

This report was authored by Dr. Jaynarayan H. Lala. Dr. T. Basil Smith was the project engineer. Mr. Charles Meisner was the NASA technical monitor for the period January-December 1982, and Mr. Nicholas Murray was the technical monitor from August 1978 to December 1981. Following are some of the people who contributed to the success of this project.

Draper Laboratory

Dr. Albert Hopkins
Mr. Jack McKenna
Ms. Linda Alger
Mr. Kevin Koch
Mr. Alan Wimmergren
Mr. Robert Scott
Mr. Joseph Marino
Mr. David Hauger
Mr. Mario Santarelli

Collins Avionics

Mr. Ron Coffin
Mr. Charles Schulz

This Page Intentionally Left Blank

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This report is Volume III of a multi-volume report on the Fault-Tolerant Multiprocessor (FTMP) project sponsored by the Langley Research Center of the National Aeronautics and Space Administration under Contract NAS1-15336. The major topic covered by this volume is the test and evaluation of the FTMP. A prerequisite for understanding this report is some knowledge of the FTMP architecture and its principles of operation described in Volume I and the FTMP Executive software described in Volume II.

The reliability, performance, and availability of the FTMP have been modeled extensively (1,2). A number of assumptions were made about various FTMP characteristics to arrive at these models. Some of these assumptions, such as mean time between failures of a line replaceable unit (LRU) can only be verified by fielding the equipment and observing its failure rate in its real operating environment. Other assumptions, though, are much more easily verified in a laboratory environment. Examples of these are the mean and distribution of the time to recover from faults and the resiliency to single point failures. FTMP response to faults can be observed and measured much more accurately under

controlled laboratory conditions rather than in the field. This is one of the motivating factors that led to a series of experiments in which the FTMP was subjected to numerous artificially created faults while operating routinely in a simulated aircraft environment and its response in each case was observed and recorded.

Apart from verifying modeling assumptions, there are a number of other important reasons for experimental test and evaluation. These include expanding validation envelope, building confidence in the system, revealing any weaknesses in the architectural concepts and/or their execution in hardware and software. Other benefits of the test and evaluation exercise include a general stressing and shake-out of the hardware as well as software, in particular, the fault detection hardware and the fault identification and system configuration control software. The results of these experiments, therefore, not only include hard data such as fault detection and identification times but a number of intangibles as well. These are discussed in Chapter 4.

The goal of the fault-injection experiments was to inject at least stuck-at-0 and stuck-at-1 class of faults on every circuit pin of one LRU and measure the FTMP response. It will be recalled here that all ten LRUs in the FTMP are identical to each other in hardware. In addition, due to the symmetric architecture of the multiprocessor and the Executive, the functions performed by one LRU over a period of time are no different from those performed by any other LRU that is operational and active. Therefore, one can be fairly confident in assuming that the results obtained by subjecting only one out of ten LRUs in the system to

faults are representative of what would be observed if the faults were distributed amongst all ten hardware units. The choice of stuck-at class of faults was necessitated by limited time and resources rather than any deficiency in the experimental set-up. The fault injector, to be described in the next chapter, is in fact fully capable of generating and injecting a wide variety of faults including externally supplied signals. The fault injector can simultaneously produce 48 fault signals, each of which could conceivably be applied to a different circuit pin simultaneously. Once again, due to the previously mentioned limitations and the astronomically high combinations of multiple faults (even just double faults) as well as the extremely low probability of such events ever happening in real life, it was decided to limit the experiments to a single fault at a time.

The FTMP response was measured in terms of fault detection, isolation, and recovery times. Identity of the faulty unit, as determined by the multiprocessor, was also recorded.

It was determined early in the experiments that once the fault had been detected it took a deterministic amount of time to identify the faulty unit and to reconfigure the system such that the faulty unit was no longer active. The fault detection time, on the other hand, was found to be quite variable for a given fault on a given pin. This variation was dependent on when the fault was injected with respect to the internal FTMP frames. To reflect this variation in the experimental data as well as to gain a measure of repeatability of FTMP performance, each fault on each pin was repeated five times. The moment at which each fault was inserted was randomized with respect to the basic FTMP software cycle.

The next chapter describes the customized fault-injection hardware and software and the experimental set-up. Results of the experiments are discussed in Chapter 3 and the last chapter summarizes the conclusions.

CHAPTER 2

EXPERIMENTAL TECHNIQUES

2.1 Overall Experimental Set-Up

To inject faults into the FTMP, a device called the fault injector (FI) was designed and built at CSDL. The fault injector interfaces with the FTMP on one end and with the PDP-11/60 Unibus on the other end. The number and type of faults and the time of their insertion are controlled by Fault-Injection Software (FIS) that is resident in the PDP computer. The PDP-11 and the FTMP are linked together by a MIL-STD 1553 bus. This data bus is used by the Fault-Injection Software to communicate with the System Configuration Control (SCC) task in the FTMP. This then makes the experimental set-up a closed loop system in which the executor, that is, the FIS, and the victim, that is, the FTMP, are in constant touch with each other. This, as shall be seen later, makes it possible to automate the fault-injection process and to collect data that otherwise would not be possible to acquire.

Figure 1 shows a block diagram of the experimental set-up. The victim LRU is in the upper right hand corner of the FTMP cabinet. This is LRU 3. Access to the electronic components in this unit is provided by opening the swing down door on the FTMP cabinet. This exposes the LRU

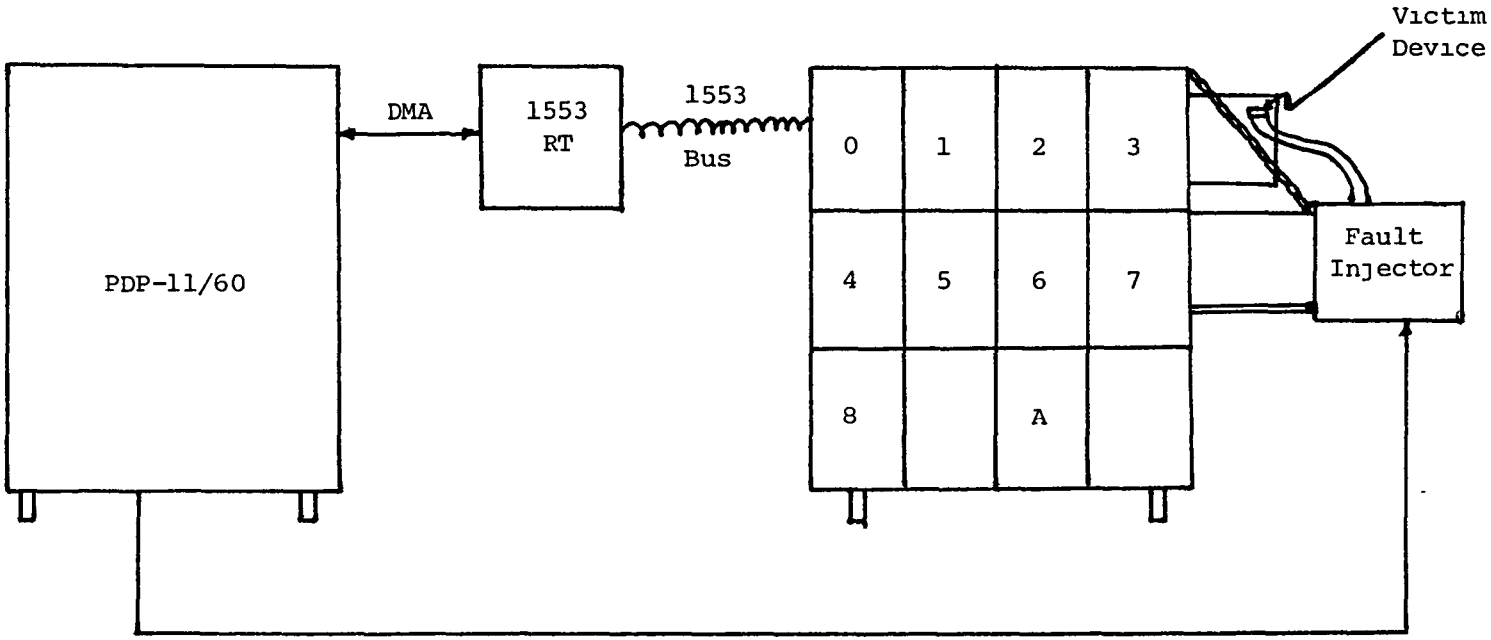

Figure 1. Fault Injection Experimental Set-Up

circuit boards which may then be extended for fault insertion. Faults are normally injected on one pin at a time. To insert faults, controllable DIP extenders or implants (part of the fault injector) are plugged into the DIP socket. Each implant accepts the DIP pins it replaced and contains circuitry which can interrupt and/or reconnect each DIP pin and each incident signal line from the socket. Six implants, each of which handles 8 DIP pins, are provided. Thus up to 48 pins on one DIP or on a combination of DIPs may be set up for fault injection at a given time. The standard circuit boards in the FTMP are multi-layer printed circuit boards on which the DIPs have been soldered. However, to facilitate removal of DIPs for fault injection, one complete set of circuit boards for one LRU has been furnished with DIP sockets.

The 48 implant pins of the fault injector are individually addressable by the PDP-11. Each pin appears as a Unibus address to the fault-injection software. The type of fault to be produced at any pin is controlled by writing appropriate data to the Unibus address corresponding to this pin. Once a fault or faults have been defined, they can be 'enabled,' that is, inserted into the victim by writing to another Unibus address. The fault injector hardware listens to this address space, decodes the data, and produces the fault that is called for. It also enables or clears the fault when appropriate data is written to the enable/clear address.

It is possible to produce signals other than simply stuck-at class of faults. Faults that are boolean functions of signals on other pins can be generated. This can be used to simulate faults which are rather

unlikely but which have been known to have happened. For example, it is possible to turn a NAND gate into a NOR gate. But the main utility of this capability lies in being able to inject faults into tristate signals. For example, the data pins of a random access memory have signals that are either inputs to or outputs from the memory depending on whether memory is being written to or read from. To inject a fault into such a device pin, the direction of the fault signal should be correct in order to avoid any possible damage to the device. Such a signal can be produced by generating the fault as a function of other signals on the device that determine the direction of the data such as read/write and chip enable signals on the RAM DIP.

The fault injection software has been written to facilitate automatic fault injection by providing commands that are used to define the victim device, map its pins into implant pins, define type of fault for each pin, and enable and clear faults. The FIS can execute a string of such commands, making it possible to go through a number of faults automatically once the victim device has been moved to the implants physically. A second condition necessary for automatic fault injection is some form of communication between FIS and the FTMP to indicate whether the FTMP is ready to accept a new fault. Messages between FIS and the multiprocessor are exchanged over a 1553 data bus. A modified version of the system configuration control program, called the FSCC, is responsible on the FTMP side for this protocol. Messages from FSCC are sent through I/O port 0 or 1 over the 1553 bus to a 1553 remote terminal simulator. The RT DMAs the messages into the PDP-11 memory, which then can be

accessed by the FIS program. The same data path is used in reverse to send messages from FIS to FSCC. Value of the FTMP real-time clock at the time of fault detection, identification, and recovery is recorded in the FTMP and sent to FIS. Since the time of fault injection is known to FIS, the difference between the times of fault detection and injection constitutes the time taken to detect the fault. This along with the identification and reconfiguration times and their sum are recorded in the PDP-11 for later analysis.

The fault injector hardware, fault injector software, and FSCC are described in the next three subsections.

2.2 Fault Injector Hardware

A functional block diagram of the fault injector is shown in Figure 2. The heart of the fault injector is a pair of FETs that is interposed between the device pin and the socket pin. By turning the FETs on, a direct connection is established between the device and the socket. This is the normal situation when no fault is being injected on the pin. The device-socket connection can be severed by turning the FETs off. Now any desired signal may be applied to the device or the socket pin, whichever pin has the input signal (see Figure 3). A choice of eight signals is provided, one of which may be selected by multiplexer M1 for the device pin and by multiplexer M2 for the socket pin, as shown in Figure 2. A set of 48 FET and mux pairs are provided, one pair for each victim pin. This allows one to extend up to 48 pins on one DIP or a combination of DIPs. The choice of faults for each circuit pin is as follows.

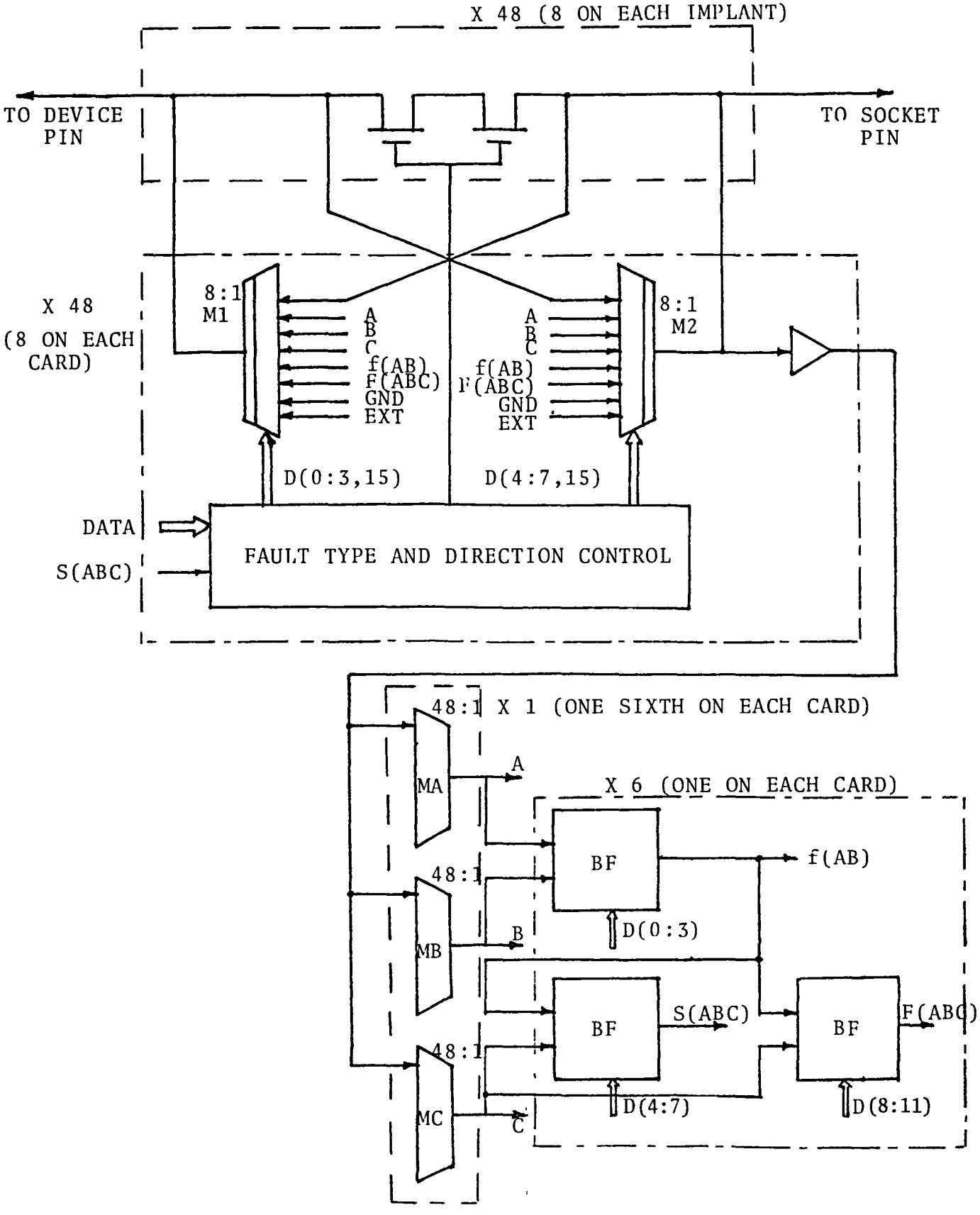

Figure 2. Fault Injector Logical Organization

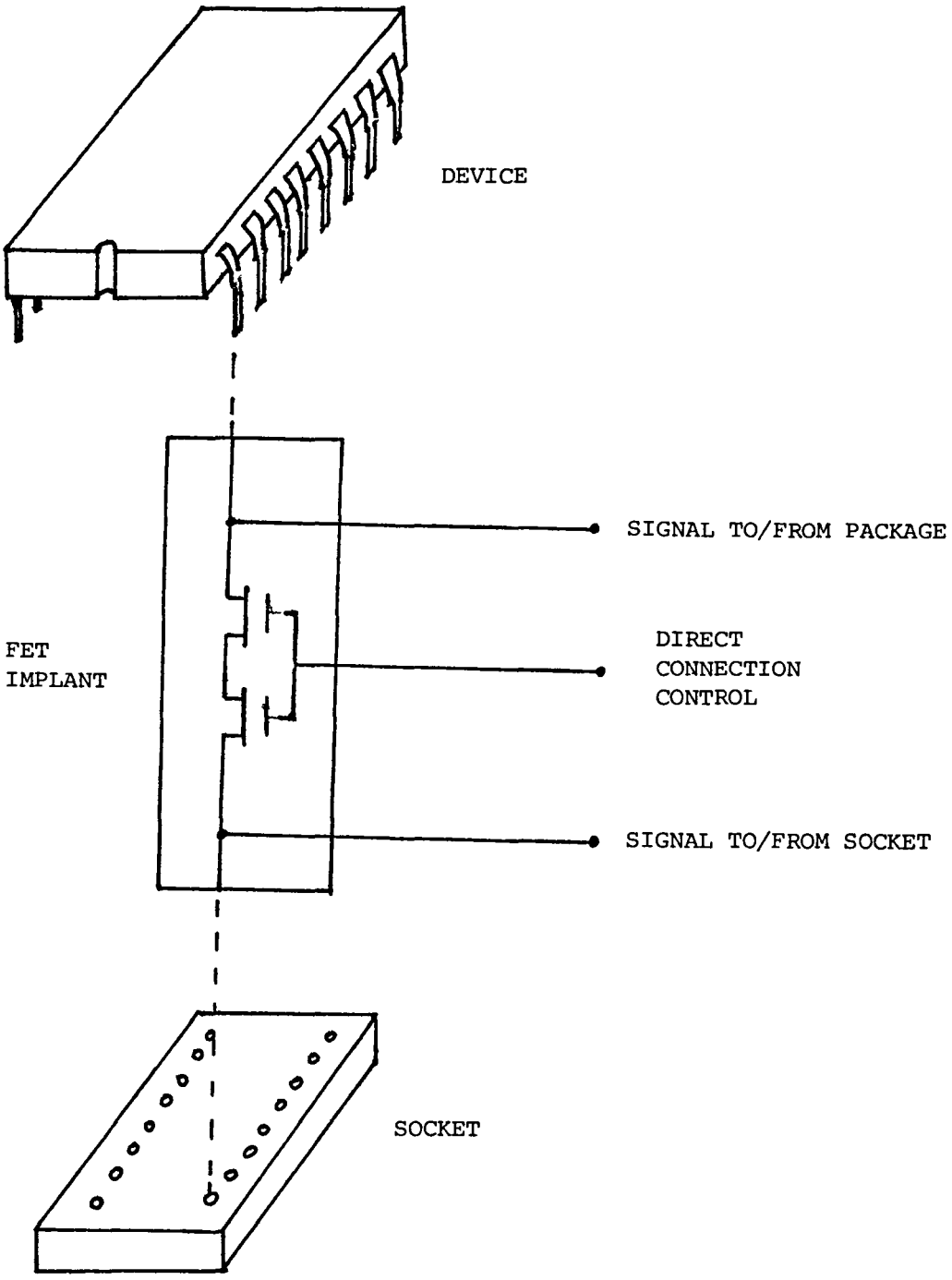

Figure 3. Insertion of FETs between Socket and Device

1. Socket/Device Signal: This provides the original signal to the victim pin. That is, no fault is injected.

2. Mux A: This signal is the output of the multiplexer A as shown in Figure 2. The inputs to the multiplexer are the 48 signals from the 48 pins that can be extended with the FETs. That is, a signal from any circuit pin or gate may be used as the fault or input signal for the victim pin.

3. Mux B: This multiplexer has the same function as Mux A.

4. Mux C: This multiplexer also has the same function as Mux A.

5. f(AB): This signal is a boolean function of two signals, the outputs of multiplexers A and B. Any one of sixteen possible boolean functions may be specified.

6. F(A,B,C): This is a boolean function of f(A,B) and the output of Mux C. Any one of sixteen possible boolean functions may be specified.

7. Ground: This provides the stuck-at-0 fault.

8. EXT: In addition to these seven choices, an externally generated signal may be used as a fault.

Each of the above eight signals may also be inverted before being applied to the victim pin, thus providing a choice of sixteen faults. The choice of faults thus includes stuck-at-1 and 'complemented signal' type of faults.

Multiplexers A, B, and C and the boolean function generators provide an extremely powerful capability to generate any type of fault. For example, certain faults in integrated circuits can change a NAND gate

into a NOR gate. It is possible with this fault injector to simulate such a fault by extending all the input and output pins of the target gate with the FETs, generating the required boolean function using inputs from the gate inputs and replacing the gate output with this signal. The main utility of this powerful capability, however, lies in the ability to inject faults on tristate signal lines. The direction of the fault can be made a function of other signals on the device, signals that determine the state of the tristate pin. It is thus possible to inject faults into the data pins of memory chips and other tristate devices.

The fault injector hardware is physically packaged as shown in Figure 4. The FET pairs are mounted on an implant segment. Two sizes of implants are provided: 4 pin extenders and 8 pin extenders. An 8 pin implant has 16 FETs mounted on it and can extend one side of a 16 pin DIP. Dummy extenders that simply connect socket and device pins without going through a FET are also provided. These are used to extend those device pins that are too sensitive to sustain the capacitance and/or time delay of an intervening FET.

In any event, the FET implants are connected to multiplexer boards through a flat ribbon cable. As mentioned earlier, the fault injector has the capability of extending 48 device pins. Signal on each of these pins is controlled by a dedicated pair of multiplexers M1 and M2 (see Figure 2). Thus there are a total of 48 pairs of muxes. These are packaged on six multiplexer boards as shown in Figure 4. Each board controls 8 pins. One 8 pin implant or two 4 pin implants may be connected to each board. The six boards, labeled A, B, C, D, E, and F, are

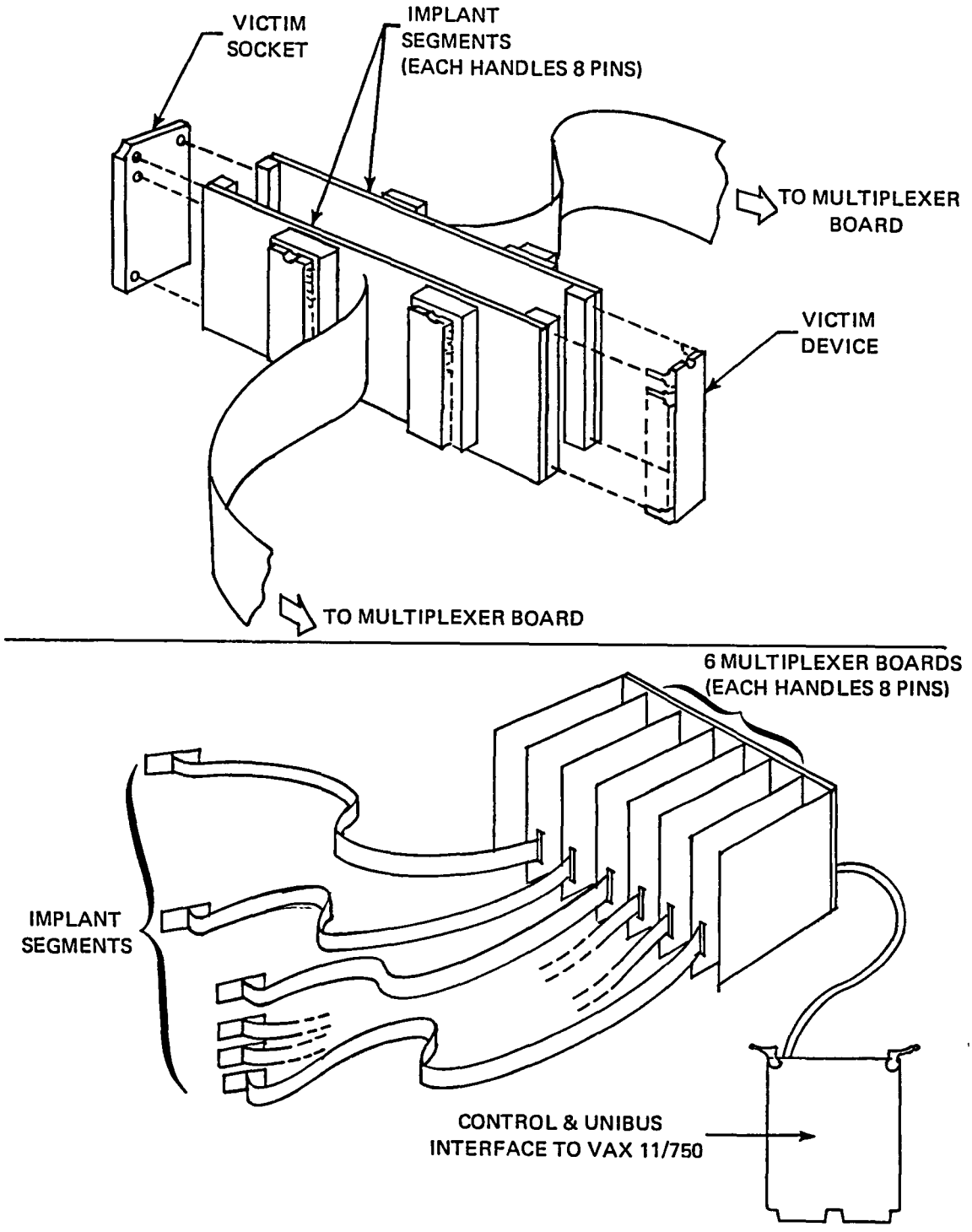

Figure 4. Fault Injector Hardware

identical multiwire boards. Each of them also contains one sixth of the multiplexers MA, MB, and MC. That is, each of the three 48:1 muxes (A, B, and C) is logically partitioned into six 8:1 muxes. Since a board handles 8 pins, a signal from these eight pins can be selected through the 8:1 mux (A, B, or C) on that board. The outputs of six logical parts of each 48:1 mux are OR'ed and distributed to all six circuit cards via the backplane. All 48 signals are then made available to each board. Each board also has its own copy of the three boolean function generators shown in Figure 2. Functions f(A,B), F(A,B,C), and S(A,B,C) can be produced on any board. These signals, along with outputs of muxes MA, MB, and MC form inputs to the muxes M1 and M2.

Last, but not least, is the selection and control of FETs, multiplexers and boolean function generators, and enabling and clearing of faults. The fault injector has been designed such that it can be addressed as a unibus device by a PDP-11 or VAX-11 computer. The data written to the unibus address space of the fault injector is used to perform the selection and control functions. As shown in Figure 4, the backplane of the multiplexer boards is connected by four flat-ribbon cables to a control and unibus interface card. This is a double-height wire-wrap board that can be plugged into the PDP-11 unibus. It has the standard unibus protocol and address decoding circuitry. The fault injector occupies the address space 764600-764777 (octal). This address space is mapped as shown in Table 1.

Circuitry controlling signals on each of the 48 pins (muxes M1, M2, and FETs) is addressed individually (addresses 764600 to 764736).

Table 1. Fault Injector Address Space

Address 764xxx	Mux Board	Pin
600-616	A	1 to 8
620-636	B	1 to 8
640-656	C	1 to 8
660-676	D	1 to 8
700-716	E	1 to 8
720-736	F	1 to 8
740	MUX A	
742	MUX B	
744	MUX C	
746	Boolean Function Select	
752	Execute/Clear Fault	
750 and 754-777	UNUSED	

Data written to these addresses selects one of eight inputs to mux 1 or 2 and controls the point of fault insertion (device or socket) by choosing mux M1 or M2. This is a static operation. That is, the data written to these addresses is latched in the fault injector. The type and direction of fault signal is thus determined, but the signal is not applied to the victim yet. To actually break the device-socket connection and inject the fault signal, one must write to Execute/Clear address 764752.

Writing a "0001" to the address enables the chosen multiplexer M1 or M2 on the chosen pin as well as turns the pair of FETs on that pin off. Faults on all the previously "enabled" pins are asserted simultaneously. The most significant bit of the fault selection data word determines if a pin is enabled. Writing a "0002" to the Execute/Clear address disables the muxes and turns the FETs on, thus clearing the fault condition.

Bits 0-3 of the data word select the type of fault going to the device pin, bits 4-7 select the fault going to the socket pin, and bits 8-11 determine the direction of the fault (to device or to socket) as shown below in Figure 5.

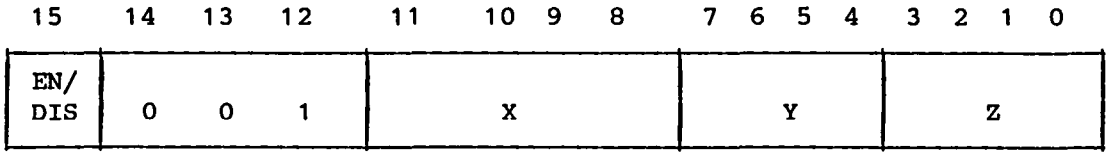

Figure 5. Fault Description Word

Bit 15 enables/disables the pin. A pin must be enabled before a fault defined on it can be asserted. Bit 15 must be 1 for the pin to be enabled. Bits 12, 13, and 14 should always be as shown in Figure 5.

If fault direction field is 0, the fault as determined by data bits Y is sent to socket pin and data bits Z are ignored. If X is 8, the fault as determined by Z is sent to device pin and Y is ignored. In addition to 0 and 8, there are fourteen other values that can be assigned to X. Fault direction selected for these values of X is explained later in this section.

Table 2. Fault Type Selection

Y/Z	Fault Signal
0	Inverted Signal
1	F(A,B,C)
2	A
3	B
4	C
5	f(A,B)
6	1
7	EXT
8	Original Signal
9	$\overline{F(A,B,C)}$
A	$\overline{A}$
B	$\overline{B}$
C	$\overline{C}$
D	$\overline{f(A,B)}$
E	0
F	$\overline{EXT}$

Y and Z select the fault signal as shown in Table 2. If Y/Z is 8, the original signal is passed through the multiplexer unchanged. Stuck-at-1 and 0 faults can be generated by a value of 6 and "E," respectively. The signal can be inverted if Y/Z is 0. Other more complex faults can be chosen as outputs of multiplexers A, B, C or a boolean function of their outputs (Y/Z = 1 to 5, 9 to "D").

If a multiplexer A, B, or C output is either used directly as a fault or as input to a boolean function generator, it is necessary to

select the multiplexer source. This is done by writing to the unibus address of the multiplexer. Data written to multiplexer address is interpreted as shown in Figure 6.

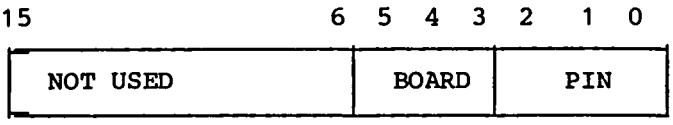

Figure 6. Mux A, B, C Selection Word

Bits 3, 4, 5 select one of six boards A to F. Bits 0, 1, 2 select one of eight pins on that board as the mux output. These are shown in Table 3.

Table 3. Mux A, B, C Source Selection

Data Bits 3 4 5	Board Selected	Bits 0 1 2	Pin Selected
1	A	0	1
2	B	1	2
3	C	2	3
4	D	3	4
5	E	4	5
6	F	5	6
		6	7
		7	8

If a boolean function such as f(A,B) or F(A,B,C) is chosen as the desired fault, then one must also define the boolean function by writing

to function select address 764746 (octal). The data written to this address is interpreted as shown in Figure 7.

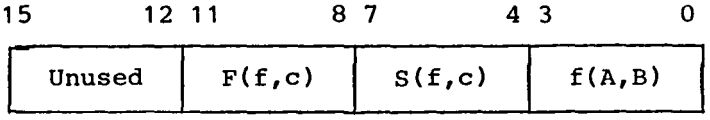

Figure 7. Boolean Function Generator Data Word

Bits 0-3 are used to select one of sixteen boolean functions of signals A and B. Bits 4-7 are used to select F(A,B,C), which is one of sixteen boolean functions of f(A,B) and C. Bits 8-11 are used to select S(A,B,C), which is also one of sixteen boolean functions of f(A,B) and C. The sixteen possible boolean functions of two variables are shown in Table 4.

As noted earlier, the fault direction (to device or to socket) is controlled by a 4-bit field X as shown in Figure 5. X can assume one of 16 possible values. These are interpreted as shown in Table 5.

If the fault direction signal chosen by X is high the fault is asserted on a socket pin. If it is low, the fault is asserted on a device pin.

For X equal to 0 the fault direction signal is high and the fault is sent to socket. For X equal to 8 the fault is applied to device. The fault direction in these two cases is static. For other values of X the fault would be dynamically applied to the socket or device pin depending upon whether the chosen signal is high or low, respectively. The signals

Table 4. Boolean Functions of Two Variables

Data	Boolean Function of A, B
0	ϕ
1	$\overline{A} \cdot \overline{B}$
2	$A\overline{B}$
3	$\overline{B}$
4	$\overline{A}B$
5	$\overline{A}$
6	$A + B$
7	$\overline{A} + \overline{B}$
8	AB
9	$\overline{A + B}$
10	A
11	$A + \overline{B}$
12	B
13	$\overline{A} + B$
14	$A + B$
15	1

that can be used for direction control are the outputs of multiplexers A, B, C, or their boolean functions f(A,B), S(A,B,C) and their complements. This allows one to dynamically control fault direction on tristate pins.

As explained earlier, fields Y and Z in the fault description word determine the type of fault to be applied to socket and device pins, respectively (see Figure 5). When X is equal to ϕ or 8 only one of these two fields (Y when X is 0 and Z when X is 8) need be defined. However,

Table 5. Fault Direction Control

X	Fault Direction
0	TO SOCKET
1	$\overline{S(A,B,C)}$
2	$\overline{A}$
3	$\overline{B}$
4	$\overline{C}$
5	$\overline{f(A,B)}$
6	NOT USED
7	NOT USED
8	TO DEVICE
9	S(A,B,C)
A	A
B	B
C	C
D	f(A,B)
E	NOT USED
F	NOT USED

both Y and Z must be defined when x is not 0 or 8. But Y and Z need not be the same. That is, different faults can be applied to socket and device pins. In fact, by an appropriate choice of Y and Z a fault can be applied in one direction while the original signal is passed through unchanged in the other direction. One may, for example, wish to insert a fault in a data pin of a memory chip only when data is being read out but not when data is being written into the memory. This can be done by selecting a fault direction signal that is high during the memory read

cycle. The fault selected by Y would be applied to the socket pin during the read cycle. By choosing Z to be 8, correct data would be written to the memory during memory write cycle since Z = 8 passes the original signal to the device pin.

Multiplexer output selection and boolean function definition are static functions. Data written to these addresses is latched in the fault injector.

It should be mentioned here that the fault injector is a 'write-only' device. State of the fault injector cannot be determined by reading its address space.

It is not necessary to remember various addresses of the fault injector since the fault injector software maintains these tables as a data base. FIS provides appropriate commands to define fault types and select mux outputs. The next subsection describes the fault injection software.

2.3 Fault Injection Software

The fault injection software (FIS) package resident on the PDP-11 provides commands at a PDP-11 terminal to perform all the functions necessary to inject faults into LRU 3 of the FTMP and observe and record the results. The FIS program is invoked by the command FIS. Valid FIS commands and their functions are as follows:

DEFINE Unn M: This command defines an M pin IC package whose location on the FTMP circuit board is Unn. Last package so defined becomes the 'active' package.

MAP n AM ℓ: This maps pin n of the active package into pin m of the multiplexer board A of the fault injector. ℓ-1 subsequent device pins are mapped to ℓ-1 subsequent board A pins. Device pins may be similarly mapped into pins of boards B, C, D, E, and F by substituting the appropriate letter in place of A in this command. This mapping allows one to reference device pins directly in subsequent commands. This mapping is stored as one of the FIS data bases.

DESCRIBE n abcd: This command defines the fault (abcd) to be injected into pin n of the active package. abcd is a 16-bit hexadecimal number that defines the fault as shown earlier in Figure 5 and Table 2. No mnemonics are provided to define the fault type and one must consult this table to create the fault selection data word. The FIS program converts the device pin number into the implant address using the previously defined pin mapping data base and the fault injector address data base. The data word abcd is then written to this unibus address. The data is latched in the fault injector hardware but the selected fault is not yet asserted.

SELECT Packagename: Subsequent MAP, DESCRIBE, and ENABLE commands refer to the selected package.

MUX n Unn m: This command is used to select pin m of package Unn as the output of the multiplexer A, B, or C depending on whether n is 1, 2 or 3. Valid values for m are 1 to 48. The FIS program maps the package pin in question into a board and pin number and formats an appropriate data word as defined in Figure 6 and Table 3 of the previous section. This data word is then written to the unibus address corresponding to the selected multiplexer.

FUNC abcd: This command is used to select the boolean function. One must consult Figure 7 and Table 4 to construct the function select word abcd. This command simply writes this word into the function select address.

ENABLE n: This enables or selects pin n of the active package. A pin must be enabled before a fault can be asserted on it. The FIS program enables the pin by writing the fault selection data word (previously defined for this pin) OR'ed into "8000" (hex), that is, with the enable/disable bit turned on. It will be recalled here that the fault injector hardware is a 'write-only' device. Therefore, a shadow of all faults previously defined by Describe commands is maintained as an FIS data base.

DISABLE n: This disables or deletes pin n of the active package. This is done by writing the fault selection data word previously defined for this pin with the enable/disable bit turned off.

DUMP: This command is used to dump on the terminal the fault description, mapping and enable/disable status of each of the 48 pins of the fault injector.

EXEC: This command actually injects or asserts faults on those pins that have been enabled. This is done by writing 1 to the Execute/Clear address. Ten seconds later the fault condition is cleared by writing 2 to the same address.

AUTO n: This command repeats the EXEC function n times. However, before injecting a fault, a 'Get Ready' command is sent by FIS to FSCC program in the FTMP. The system configuration controller in response to the command checks the status of LRU 3 and brings it on-line if they are

not already active. An 'I am Ready' signal is sent back by FSCC to FIS. The FIS program waits for a random time between 0 and 999 msec before inserting the fault. This allows the fault insertion time to be sufficiently randomized with respect to the FSCC task which is also responsible for detecting faults in the FTMP.

OUTPUT filename: This command saves the results of the fault injection experiments in the specified file. The results consist of fault detection, isolation, and reconfiguration times and the total recovery time, that is, the sum of the FDIR times.

The core of the FIS program is written in FORTRAN IV PLUS. It uses the line parser provided by the RSX-11M operating system to interpret the commands described above. Once a valid command has been identified, appropriate subroutines are called to perform the required function. This may involve updating its data base such as that required by DEFINE and MAP commands or it may require computing a unibus address by consulting its data base and writing data to this address. An assembly language subroutine actually does the I/O. The FIS program also communicates with the FSCC task in FTMP in response to the AUTO command. The FIS-FSCC protocol is described in the next section.

EXIT: This command is used to exit from FIS program.

2.4 FSCC

FSCC is a version of the System Configuration Control (SCC) task in the FTMP that has specifically been modified to work with the FIS program in the PDP-11. It is assumed here that the reader is familiar with the contents of Volume II which describes the basic SCC program in detail.

There are two major differences between SCC and FSCC. First, FSCC does not cycle spare processors, memories, or buses into active state. It maintains a fixed system configuration under normal circumstances. Of course, if it detects a fault it would try to identify the faulty unit and reconfigure it out of the system. Second, by communicating with FIS it ensures that the victim LRU, that is, LRU 3, is active before FIS inserts a fault into one of the LRU pins. The FSCC-FIS protocol works as follows:

When FIS is ready to inject a fault, it sends a 'Get Ready' message or command word to FSCC. FSCC looks at this word in its normal mode. If it is true, the FSCC state is changed to 'Reconfigure' and the reconfiguration state is initialized to 13. Recall that SCC state 13 corresponds to cycling spare units. In FSCC spares are not cycled. Instead in this state the status of processor 3 and memory 3 is checked. If they are failed, they are repaired by changing their status in the system configuration tables. The reconfiguration state is changed to 100 so that on the subsequent FSCC pass the spare units, viz. processor and memory 3, can be assigned to shadow active triads. If the units were not failed, the state is changed to 14. In this state, swap commands are issued to swap processor and memory 3 into active members of their parent triads. The state is changed to 15. Also, a signal called 'Acknowledge Get Ready' is sent to FIS acknowledging that the Get Ready command has been received and acted upon by FSCC. FIS then clears Get Ready. Clearing the command prevents FSCC from needlessly checking the status of LRU 3 repeatedly. FSCC stays in state 15 until swap commands have been

executed. It then sends an 'I am Ready' message to FIS indicating that LRU 3 components have been repaired and are in the active state. The detect, identify and reconfiguration times are simultaneously cleared to zero. FSCC then resumes its normal state. In this state it reads error latches and does fault detection.

After receiving the 'I am Ready' message, FIS waits for a random length of time that is uniformly distributed between 0 and 999 milliseconds. This corresponds to between 0 and 3 cycles of the FSCC task. This random wait assures that the fault is not always injected at the same time with respect to execution of the fault detection program in the FTMP.

When FSCC detects the fault it notes the value of the FTMP Real Time Clock. The clock values at the instant of fault identification and system reconfiguration are also recorded. FSCC thus has all the information to compute the time intervals between fault detection and fault identification as well as that between identification and system recovery. The identification and recovery time intervals can be computed with an accuracy equal to the least count of the Real Time Clock which is 1/4 millisecond. However, FSCC can not compute the fault detection time since it does not know when the fault was injected. To compute detection time, the FTMP time base, that is, the Real Time Clock, is sent to FIS every R4 frame. Typically, R4 rate group iteration period is 40 milliseconds. Therefore the FIS program knows the FTMP time of fault injection to within 40 milliseconds. Although this biases the detection time on the average 20 milliseconds (towards higher values), as will be

seen in the next section, this is not a significant amount of error in the overall detection time distribution. At any rate, the Real Time Clock is sent to FIS every R4 frame. The value of the RT Clock at the time the fault is detected, identified, and recovered is also sent to FIS. FIS then computes the detection, identification, and recovery time intervals and records them in a file.

The fault condition is cleared as soon as the FTMP has recovered from the fault. FIS keeps track of FTMP's progress in recovering from the fault by monitoring the FDIR times being sent to it. Recall that these locations are cleared to zero before a fault is injected. Therefore as each of these words assumes a non-zero value it shows FTMP's progress through various stages of system recovery. To assure that there is no deadlock in the FSCC-FIS protocol, a number of time-out conditions are provided. If after a predetermined time the FTMP has not detected the fault, the fault signal is removed and the FIS program proceeds to the next command line. Similar timeouts are provided for the identification and recovery phases. The length of these timeouts can be chosen when the FIS program is initially invoked.

The block of data exchanged between FIS and FSCC is as shown in Table 6. Note that the Real Time Clock as well as all other time values are two 16-bit words.

With the exception of cycling of spares and the changes in the system configuration controller described here, the rest of the software being executed by the machine while undergoing fault injection is that

Table 6. FIS-FSCC Data Exchange Block

Data	No. Words
FIS to FSCC	
Get Ready	1
FSCC to FIS	
Real Time Clock	2
Detect Time	2
Identify Time	2
Recover Time	2
Faulty Unit	1
Reason Code	1
Ack. Get Ready	1
I am Ready	1

described in Volume II of this report. This consists of the Executive, Self-Test programs, console display, autopilot and other applications code that normally runs on the FTMP.

The next chapter describes the results of the fault injection experiments.

CHAPTER 3

RESULTS

3.1 General Observations

Faults were injected in pins of eight circuit boards. These boards are CPU Data Path, CPU Control Path, Processor Read Only Memory, Processor Cache Controller, Bus Guardian Unit (A), Bus Interface (Transmit Bus), Bus Interface (Poll and Clock Buses), and System Bus Controller. Although the overall process of physically setting up each device for fault injection, selecting 'safe' pins as targets, running the experiments and acquiring data was quite tedious and time-consuming, it went rather smoothly. There were some minor difficulties encountered with some devices and circuit boards, but once past the initial learning curve these were overcome quickly. One of the irritating factors was the extreme sensitivity of some devices to being extended on an implant. Parent module of such a device would not function correctly in the presence of an intervening pair of FETs and would be discarded by the system immediately. One obviously had the choice of ignoring that device for the purposes of fault injection and moving on to another circuit. In fact, since the correct functioning of the parent module apparently is so dependent upon that device, it is evident that a fault in the target

device would be detected immediately. One may therefore not worry too much about not being able to subject such a sensitive circuit to artificially created faults. However, as it turns out the sensitivity of a device to being extended through FETs is usually limited to one or two pins only. Once these pins have been identified (a rather tedious procedure), they can be extended with dummy implants while the remaining pins on that package can be extended through FETs. This procedure was followed for most of the sensitive packages. Since no data was acquired on sensitive pins, these pins are not included in the data analysis.

Some packages were only marginally unhappy over being extended. That is, LRU 3 would work correctly with such a device moved to an implant most of the time but not all the time. The result was that the unit would occasionally be declared failed by the FTMP even before a fault was injected. This obviously produced negative fault detection time. This, however, happened very infrequently and the results presented here, of course, exclude negative detection times.

One other practical problem that prevented subjecting some boards to fault injection was the extreme caution required in handling CMOS circuit devices. The memory chips on processor cache RAM and system memory boards are all CMOS type.

A few faults were injected in the cache RAM board but soon the socketed circuit board stopped working, most likely due to inadequate care exercised in removing and inserting CMOS memory chips. The cache RAM and the two system memory cards in each LRU are all identical and only one socketed circuit board was provided for all three. No useful data was acquired for any of the three applications of this card. The

two BGU cards also contain a lot of CMOS circuitry. Only 294 faults were injected in the BGU card before it too ceased to operate correctly.

Despite all the practical problems encountered, over 20,000 faults were injected into LRU 3 of the FTMP and the results recorded. Most of the faults were concentrated in the processor region of the LRU, the CPU data and control cards, the cache controller, and the PROM. However, a number of faults (over a thousand) were also injected into the error detection and masking circuitry as well as redundancy management hardware. The hardware voters, disagreement detectors, error decode ROM, and error latches for the Poll, Transmit, and Clock buses were subjected to faults as were enable/disable discretes in the Bus Guardian Unit. Parts of the System Bus Controller were also targeted for fault insertion.

Of the 21,055 faults injected in the FTMP, 17,418 were detected. That is, 3,637 or 17.3 percent of the faults went undetected. Although these results would seem to imply that the fault detection coverage in the FTMP is only 0.83, this is not necessarily so. For, to convert the fraction of faults undetected directly into lack of coverage is not correct. One must exclude from this total those undetected faults that 'do not matter.' There are a number of faults that obviously belong to this class. For instance, if only three gates from a quad NAND package are actually used on a card, whether the fourth unused gate operates correctly or not is quite irrelevant. Faults in this gate would not be detected but do not contribute to lack of coverage. Unused gates are easy to trace. Unused signals, on the other hand, are not. Faults on these signal pins would also go undetected but once again do not really

affect coverage. The CAPS-6 processor microcode in the FTMP, for example, does not utilize all the outputs of the AMD2901 Arithmetic Logic Unit (ALU). This can be ascertained only by an exhaustive search of each and every microinstruction to make sure that the output in question is not looked at. Such a study is outside the scope of this project. Approximately 80 percent of all undetected faults, or about 3,000 faults, were either on unused gates or on signals that are always low or always high under normal circumstances. Of the remaining 20 percent undetected faults, a few were analyzed in depth and were all found to belong to the 'don't care' class. Since each pin fault is repeated five times, the number of pins in question is about 60. However, a much more thorough analysis of all the undetected faults is required before a definitive statement can be made about fault detection coverage. Further discussion here is limited only to the faults that were detected.

3.2 Average and Maximum Times

As mentioned earlier, 17,418 faults were detected. All of these faults were identified correctly and the system successfully recovered from each of these faults by purging the faulty module and replacing it with a spare or gracefully downgrading the system when no spare was available. Based on these results one could conceivably argue that the fault identification and recovery coverages are each one hundred percent as far as the detected faults are concerned. It is, of course, not possible to extrapolate this perfect record for faults that were not detected and for LRU pins that were not subjected to faults during these experiments. As mentioned earlier, detection, identification, and reconfiguration times were computed for each fault. The three phases of

recovery were also summed to give the total recovery time for each fault. These results are summarized in Tables 7 and 8. The first of these tables lists the average detection, identification, reconfiguration, and total recovery time in milliseconds for each of the eight cards. The last column in this table shows the average FDIR times for all 17,418 faults. Table 8 shows the maximum times recorded in each category for each card, also shown in milliseconds.

There are certain obvious conclusions that can be drawn from figures in these tables. Let us start with the last phase of the recovery procedure first, that is, system reconfiguration phase. This phase begins as soon as the identity of the faulty module is known. At this point in time, the System Configuration Control (SCC) task is being executed. It will be recalled here that this task runs at the lowest frequency or R1 rate group (3.125 Hz). It passes the identity of the faulty unit on to the R4 dispatcher. The R4 dispatcher running at 25 Hz issues appropriate reconfiguration commands in its prolog to remove the faulty unit from the system. The reconfiguration phase is complete as soon as the faulty unit is replaced with a spare or the system gracefully degraded in the absence of a spare. The average reconfiguration time as seen in Table 7 is between 46 (SBC) and 113 (PROM) milliseconds depending upon the type of card. That is, on the average it takes between two and three R4 frames to reconfigure the system. The average reconfiguration time for all the faults is 82 msecs or two passes of R4 dispatcher. The overall average is weighed heavily by the processor region which was the subject of most faults. The average reconfiguration times for the

Table 7. Average Times (Milliseconds)

BOARD		CPUD	CPUC	PROM	CC	BGUA	BIT	BIPC	SBC	ALL	ALL EXCEPT BGUA
# FAULTS DETECTED		7266	4761	783	3508	294	214	235	357	17418	17124
AVERAGE TIME	DETECT	312	349	589	314	36554	1920	1361	678	988	378
	IDENT	82	99	59	59	133	147	229	263	88	88
	RECONF	80	83	113	88	47	53	71	46	82	82
	TOTAL	474	532	763	462	36735	2121	1662	988	1160	549

Table 8. Maximum Times (Milliseconds)

BOARD		CPUD	CPUC	PROM	CC	BGUA	BIT	BIPC	SBC	ALL
# FAULTS		7266	4761	783	3508	294	214	235	357	17418
MAXIMUM TIME	DETECT	9137	15817	21614	8122	118437	11592	4818	17056	118437
	IDENT	1009	780	810	1204	993	813	931	1625	1625
	RECONF	289	190	242	546	115	198	243	195	546
	TOTAL	9223	16231	21757	8182	118843	11707	4887	17604	118843

processor region are higher than all others, though the variation from card to card is quite small. The maximum reconfiguration times are also higher for the processor region (CPUC, CPUD, CC, and PROM) than all others, as seen in Table 7. How long it takes to replace a faulty module with a spare is, of course, dependent on the instantaneous system configuration. For instance, if the triad containing the failed processor is being shadowed by a spare processor the reconfiguration will be done simply by swapping failed and shadow processors on the bus lines. This takes only one pass of R4 dispatcher. On the other hand, if the spare is shadowing another triad it would be necessary to retire the target triad and synchronize spare with the target triad members. This obviously takes much longer since the target triad must complete all tasks in progress before retiring. In any event, the reconfiguration process is deterministic and bounded. The maximum time in the table, 546 msecs (CC), corresponds to the scenario just described.

The recovery phase just preceding system reconfiguration is fault identification. This phase begins as soon as a fault is detected. This happens in the SCC task. It terminates as soon as the fault source is located. This also happens in the SCC task. The interval between these two events is the fault identification time. Faults may be identified simultaneously with their detection in some cases. This usually occurs when a self-test program uncovers the fault. Since diagnostic programs know which region is being tested, they can usually identify the faulty module immediately. In other cases several reconfigurations may be required to sort out the fault symptoms. The average identification time

is seen to vary from 59 (PROM and CC) to 263 (SBC) milliseconds with the system-wide average being 88 milliseconds. Since one R1 frame is 320 milliseconds, it may be concluded that most faults are identified immediately. Indeed the average time for the processor region cards is between 59 and 99 milliseconds. This is because symptoms of a failed processor appear on two buses simultaneously, the Poll bus and the Transmit bus. In most cases this combination is uniquely associated with only a single processor in the system. Most processor faults are therefore identified immediately. The question may therefore be asked as to why it even takes 60 to 90 milliseconds to look up the bus assignment tables. Actually it does not really take that long to consult the appropriate data base in the shared memory. What happens in fact is that the SCC task of being the lowest priority is interrupted by higher priority tasks. It will be recalled here that the R4 rate group tasks are executed eight times and that the R3 rate group tasks are executed four times for every iteration of R1 tasks (SCC). Hence the identification program can be interrupted many times between start and finish. The identification time is measured as the total elapsed time and not as the length of time the program is active. This is, of course, as it should be.

The maximum identification times are seen to vary between 780 (CPUC) and 1625 (SBC) milliseconds with the maximum for the processor region being 1204. This parameter, like the reconfiguration time, is deterministic and bounded. The worst case scenario here is a fault on a bus that has four memory units enabled on it. If the bus itself is

faulty, it would take four diagnostic reconfigurations to isolate the bus from all other suspects. This translates into five passes of the SCC program and corresponds to the maximum time observed during the course of the experiments.

The fault detection phase is what starts the recovery process. This is more complex than other parts of the recovery procedure. Once a fault is uncovered, the ensuing processes are quite mechanical. The uncovering of a fault is, however, considerably more involved. Clock for the detection phase starts ticking as soon as the fault is injected under the command of the fault injection software running in the PDP 11/60. Faults are usually manifested as disagreements on one or more buses. These are recorded in error latches which are read by SCC every 320 milliseconds. The detection phase terminates when SCC digests error latch outputs and determines that they indicate an 'unexpected' bus error. Recall that some bus errors may always exist such as those on an unused clock bus or on a failed bus and so on. In any event, this time interval is the detection time. As explained in Chapter 2, the time of fault injection is not known to SCC and is known to Fault Injection Software as the most recent value of the FTMP Real Time Clock which is sent to FIS every 40 milliseconds. Therefore the fault detection time as recorded in the experiments is higher than the real value anywhere from 0 to 40 milliseconds or an average of 20 milliseconds. The average detection time for all the faults from Table 6 is seen to be 988 milliseconds. Therefore, the error is only about 2 percent.

The average detection time is seen to vary from 312 milliseconds for the CPU data card to over 36 seconds for the BGU card. Now if a fault manifested itself as an error on the bus soon after it was injected, the detection time would mostly consist of latency in reading error latches. Since error latches are read every 320 milliseconds on the average, this latency should only be 160 milliseconds. The average fault detection time for the processor region is around 300 milliseconds for the CPUD, CPUC and cache controller cards and 589 milliseconds for the PROM card. This implies that on the average there is considerable latency between fault injection and error manifestation. This is mostly due to the fact that not all parts of the processor region hardware are being used all the time. This is quite obviously true of the Read Only Memory. There is a considerable fraction of the PROM that contains programs that are invoked only when an error is detected. Faults in this region of the memory would not be uncovered until another fault manifests itself. The PROM is therefore tested periodically by a check-sum program. The average latency of half a second in uncovering PROM faults is simply a reflection of how frequently the check-sum program is executed. The maximum detection time for PROM faults is over 21 seconds and is a direct function of the repetition rate of the self-test program.

The average detection times for BIT, BIPC, and SBC cards are much higher than those for the processor region because faults in the bus interface cards were concentrated mainly in the error detection and masking hardware. Faults in most of this region would not manifest themselves as bus errors under routine operation. Some faults in the

voter circuitry, for instance, are highly latent since the voter output is the same as its inputs as long as the three inputs are the same. Such a fault can only be uncovered by feeding disagreeing input streams to the voter. This is done by a self-test program. It is seen that only 200 to 300 faults were injected in the bus interface cards. All of these faults were purposefully concentrated in the error detection region to uncover any weaknesses in this area since the correct functioning of the FTMP is so critically dependent upon this hardware. The results for these cards are therefore biased towards higher values. When the remaining random logic on these cards is subjected to faults, the averages would tend to move down because faults in the random logic would be uncovered by routine operation without self-test programs.

Finally, it is quite evident from Tables 7 and 8 that the average as well as the maximum detection times for the Bus Guardian Unit are an order of magnitude higher than even those for the error detection circuitry. There is a reason for this which is as follows. The BGU card contains the redundancy management hardware. Faults were injected in the enable/disable discretes that control whether a unit is enabled or disabled on a bus. Some of these faults such as the ones that disable a unit from its active bus would be detected immediately by routine operation since a single BGU can disable a unit by itself and a lack of transmission from a unit would immediately cause errors on that bus. But most other faults such as those that enable a unit on other buses or disable a unit from buses on which it is not supposed to transmit anyway would only be detected either by a self-test program that exercises these discretes

or over the long term by routine system reconfiguration. No self-test programs have been written for the Bus Guardian Unit. Therefore almost all BGU faults were uncovered by the rotation of processors and memories on different buses and by swapping of active and spare buses. While the self-test programs complete a cycle every 13 seconds, complete cycling of all spares takes 6 minutes. This is why maximum detection time for BGU faults is almost 2 minutes. It would be even higher if the timeout limit for these experiments was increased beyond 2 minutes. Faults that may have been detected with a higher time-out limit are treated in the data analysis as undetected faults.

The high detection times for BGU faults have a tremendous impact on the system-wide average. Table 7 shows that overall average detection time is 988 milliseconds, or about one second. If the average were computed for all except BGU faults, it would be only 378 milliseconds. The BGU fault detection times can be reduced by an order of magnitude by writing diagnostic programs for it.

It should be mentioned here for the sake of clarity that although routine system reconfiguration was suppressed for faults on all other cards to facilitate a reasonable FIS-SCC protocol, it was allowed for the BGU card since this was the only way of detecting BGU faults and it did not interfere with the protocol in this case.

The sum of times for the three phases constitutes the total recovery time. This is the time from the moment the fault is injected to the point in time when the system has completely recovered. Times for the three recovery phases were summed for each fault and then the sums were averaged over all faults for each card. These averages are shown in Table 6 under the heading 'TOTAL.' The total average recovery time for a

card should obviously equal the sum of the average time for detection, identification, and reconfiguration phases. In other words, average of the sums should equal sum of the averages. This is true for all cards to within 1-2 milliseconds which is the truncation error. The maximum total recovery time for a card, on the other hand, is not necessarily the sum of the maximums for individual phases since maximum detection time need not necessarily be for a fault that also takes maximum time to be identified. The maximum recovery time is therefore simply the maximum of all sums.

It is quite evident from data presented in Tables 7 and 8 and from the discussion so far that the recovery time is dominated by the detection time for each card as well as for the system as a whole. Even the processor region, which seems to react the fastest to faults, about 65 percent of recovery time is spent uncovering a fault. Therefore recovery time characteristics are very much like those of the detection time. In particular, if the average recovery time is computed for all faults except the BGU, it is found to be 549 milliseconds or about a half second compared to about a second if it is averaged over all the faults. The meaning of this is quite clear. FTMP response to faults can be improved twofold simply by writing a few diagnostic programs.

When the FTMP reliability was computed, it was assumed that the R4, R3, and R1 rate groups would execute at 40, 20, and 5 Hz rather than 25, 12.5, and 3.125 Hz used in the experiments. The fault injection data presented so far shows a strong correlation between detection, identification, and reconfiguration times and the execution frequencies of SCC,

R4 dispatcher, and self-test programs. It may be concluded, therefore, that the average recovery time could be reduced by 37.5 percent by increasing repetition rates to original goals.

The two changes suggested here would bring the average recovery time down to 343 milliseconds which is quite close to the value (250 msecs) assumed in reliability models.

Of course, what affects the actual reliability is not only the average recovery time but also its distribution and the LRU mean time between failures (MTBF). These are discussed next.

3.3 Frequency Distributions

The fault injection data was analyzed to compute probability density function (pdf) of detection, identification, reconfiguration, and total recovery times for each card separately and for the total ensemble of 17,418 faults. Estimates of pdf's are plotted as histograms in Figures 8 to 62.

A few comments regarding the organization and plotting of data are in order here. These figures are organized by cards in the same order as the numerical results in Tables 7 and 8. Figures 8 to 13 are for the CPU data card, 14 to 18 for the CPU control card, 19 to 25 for the PROM card, and so on. A different scale is used for each parameter to show as much detail as possible. All identification time histograms use a bucket size of 100 milliseconds, and all reconfiguration plots use a bucket size of 50 milliseconds. A common scale for all detection time distributions that accommodated maximum detection times and yet showed the details was not as easy to choose. Detection times for most cards are therefore plotted

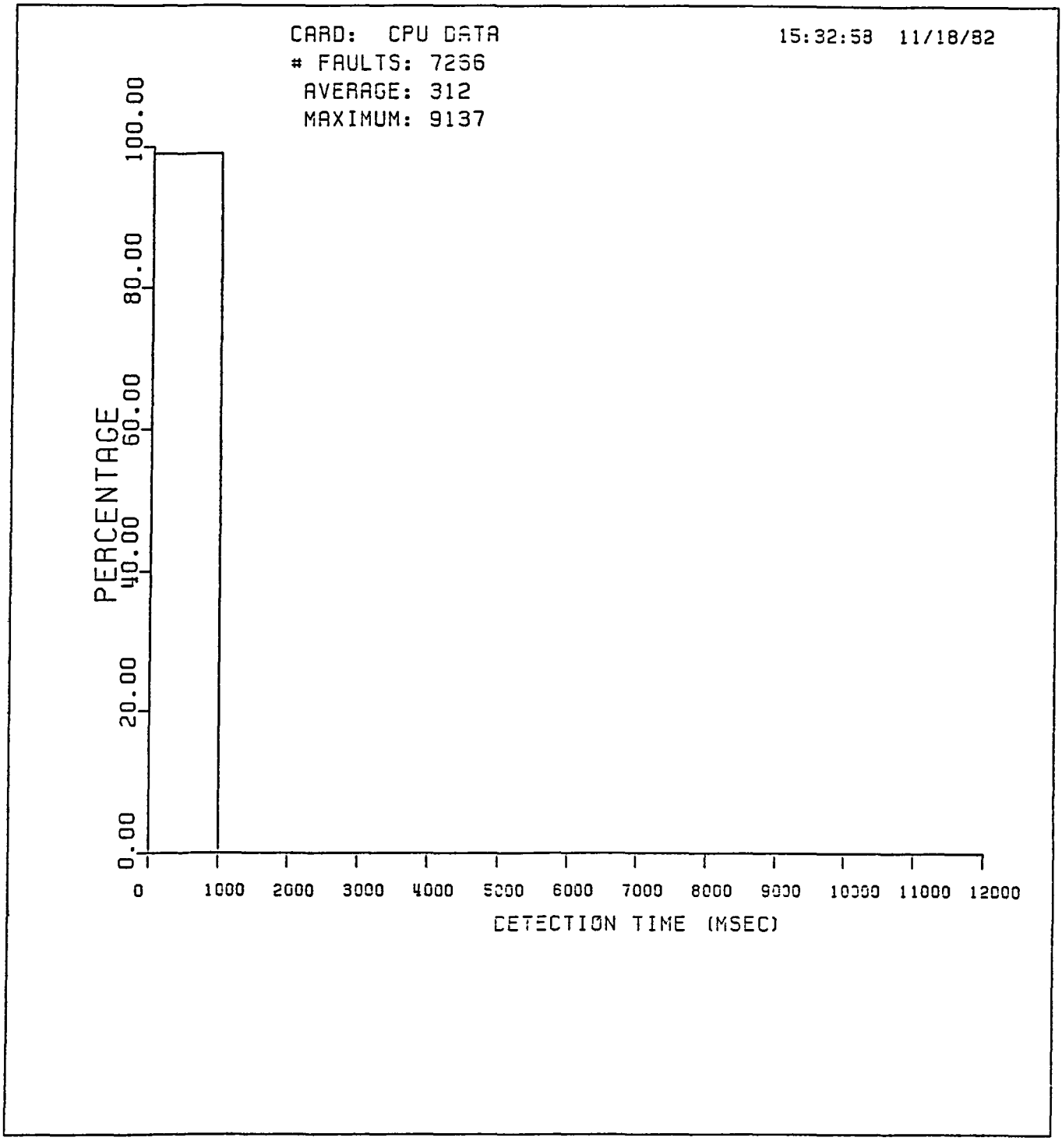
CARD: CPU DATA
15:32:58 11/18/82
FAULTS: 7266
AVERAGE: 312
MAXIMUM: 9137
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
DETECTION TIME (MSEC)

Figure 8

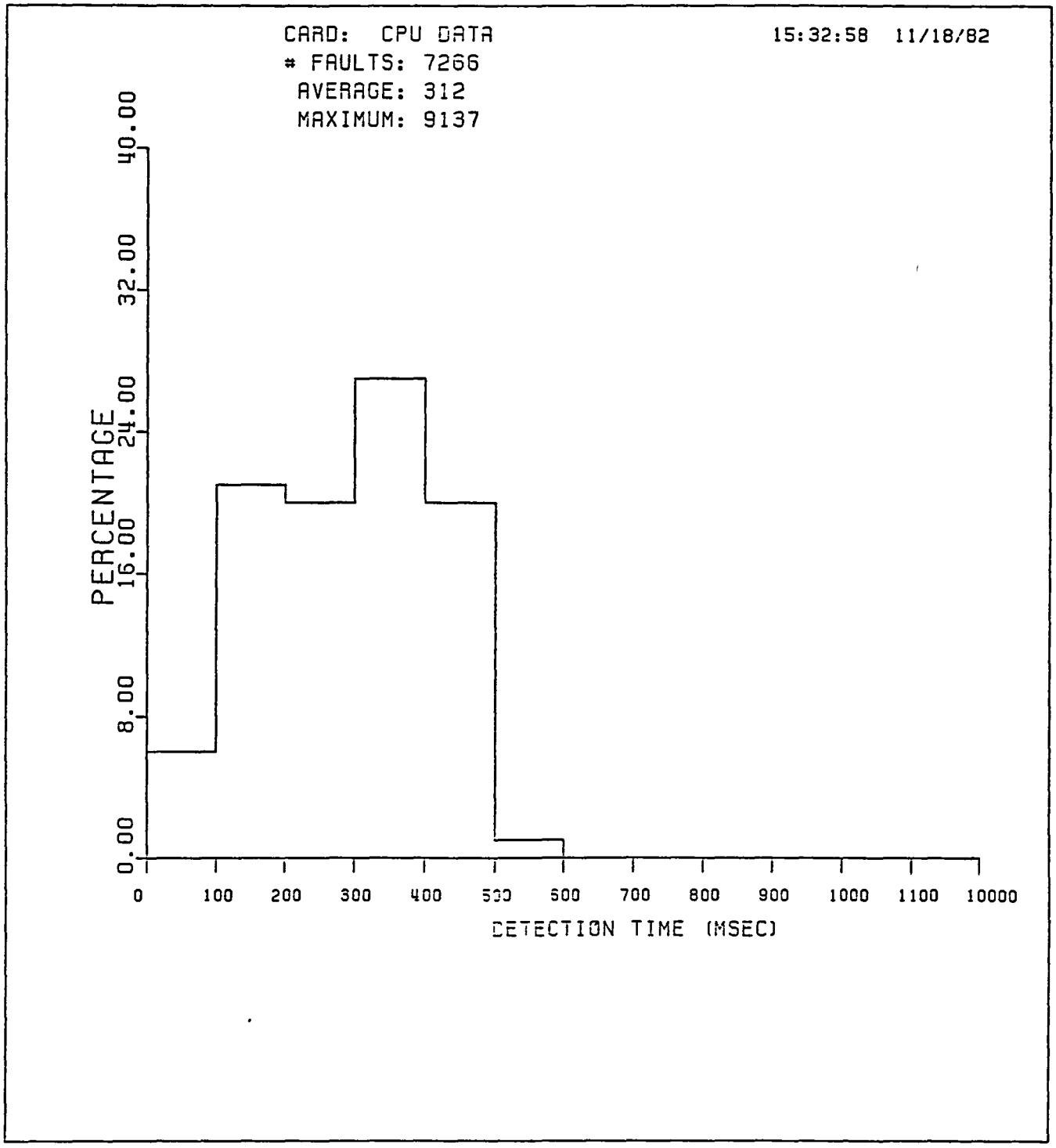
CARD: CPU DATA
15:32:58 11/18/82
FAULTS: 7266
AVERAGE: 312
MAXIMUM: 9137
PERCENTAGE
0.00
8.00
16.00
24.00
32.00
40.00
0
100
200
300
400
500
700
800
900
1000
1100
10000
DETECTION TIME (MSEC)

Figure 9

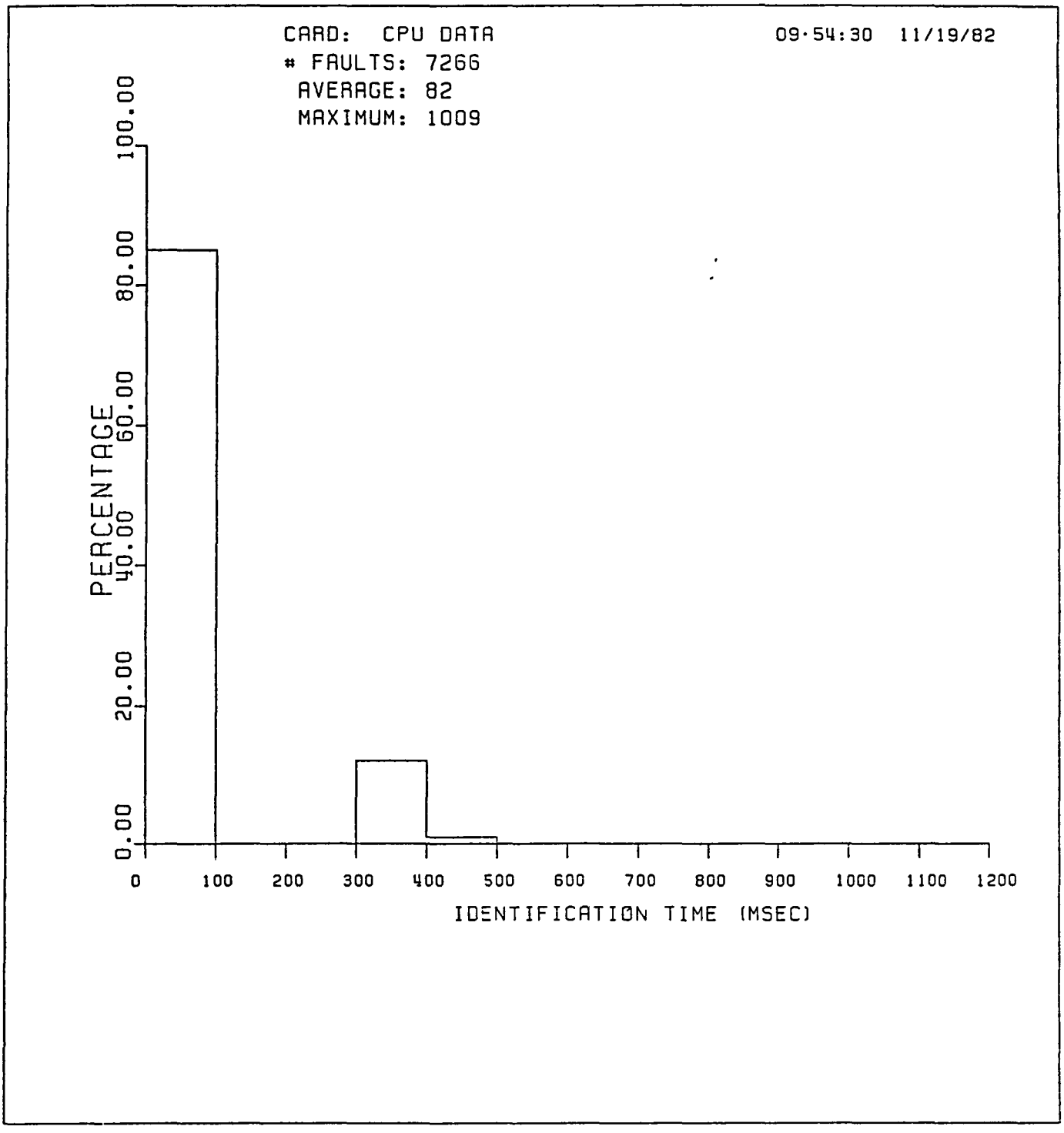
CARD: CPU DATA
09:54:30 11/19/82
FAULTS: 7266
AVERAGE: 82
MAXIMUM: 1009
PERCENTAGE
0.00
20.00
40.00
60.00
80.00
100.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 10

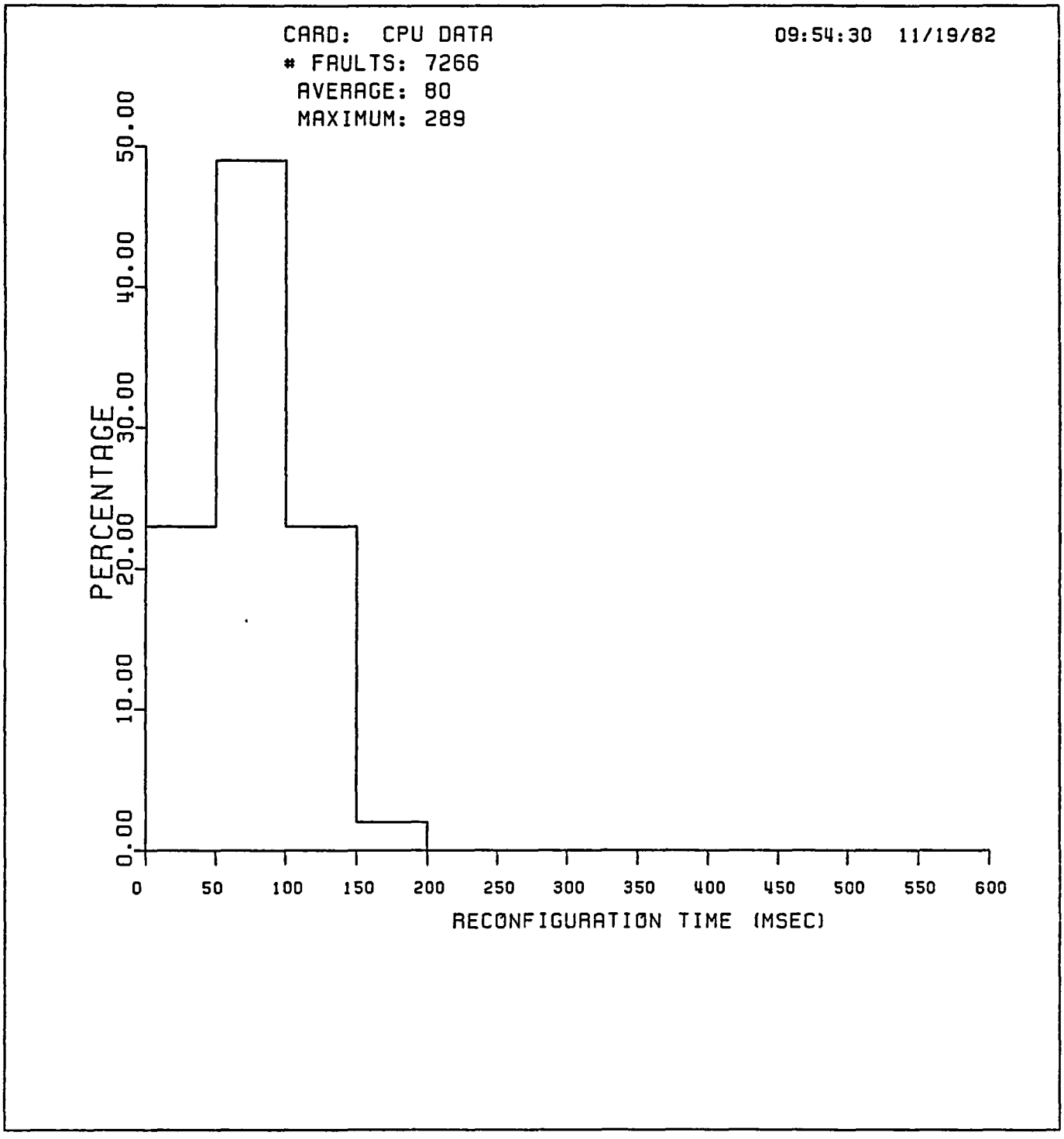
CARD: CPU DATA
09:54:30 11/19/82
FAULTS: 7266
AVERAGE: 80
MAXIMUM: 289
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 11

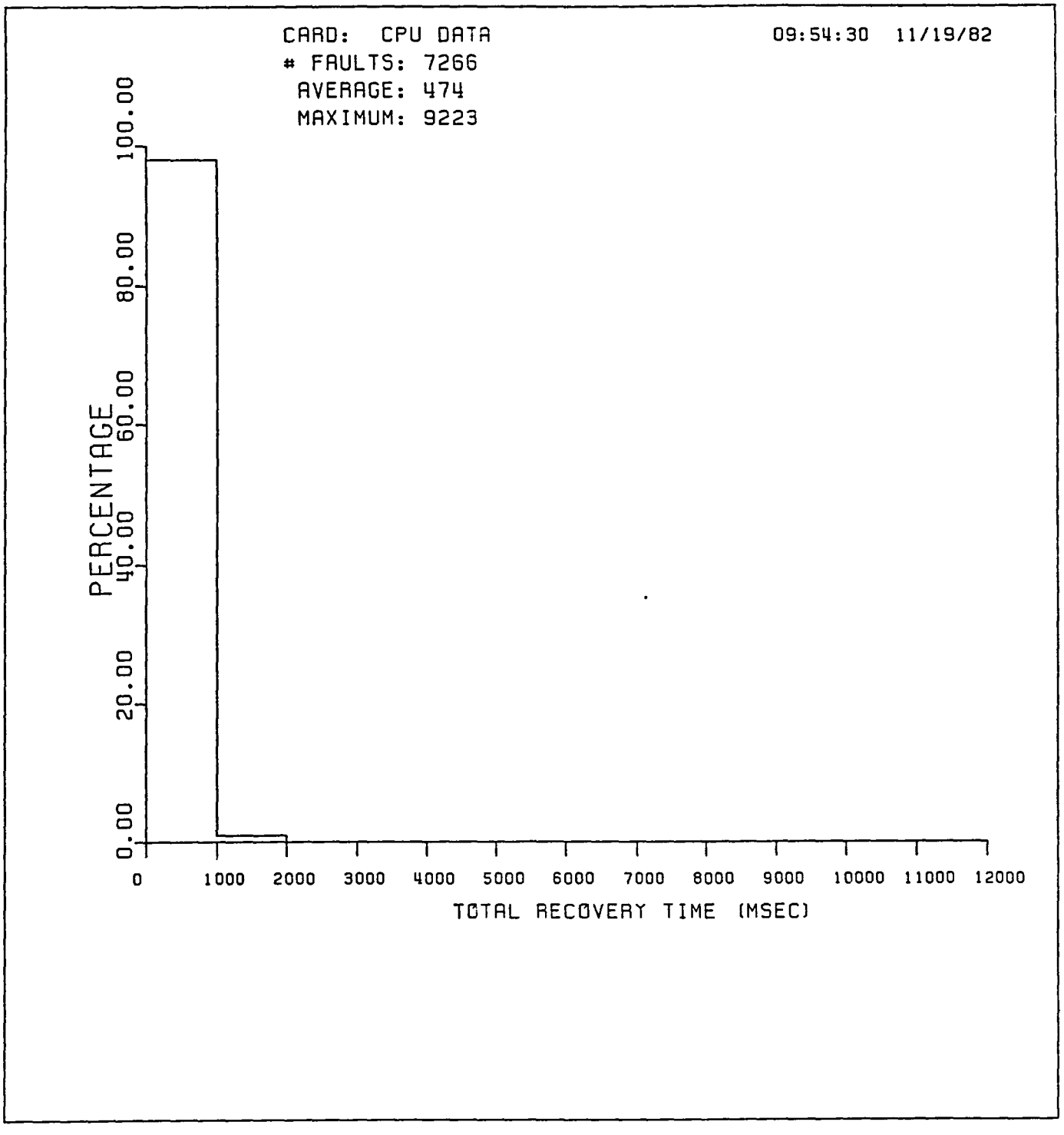
CARD: CPU DATA
09:54:30 11/19/82
FAULTS: 7266
AVERAGE: 474
MAXIMUM: 9223
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
TOTAL RECOVERY TIME (MSEC)

Figure 12

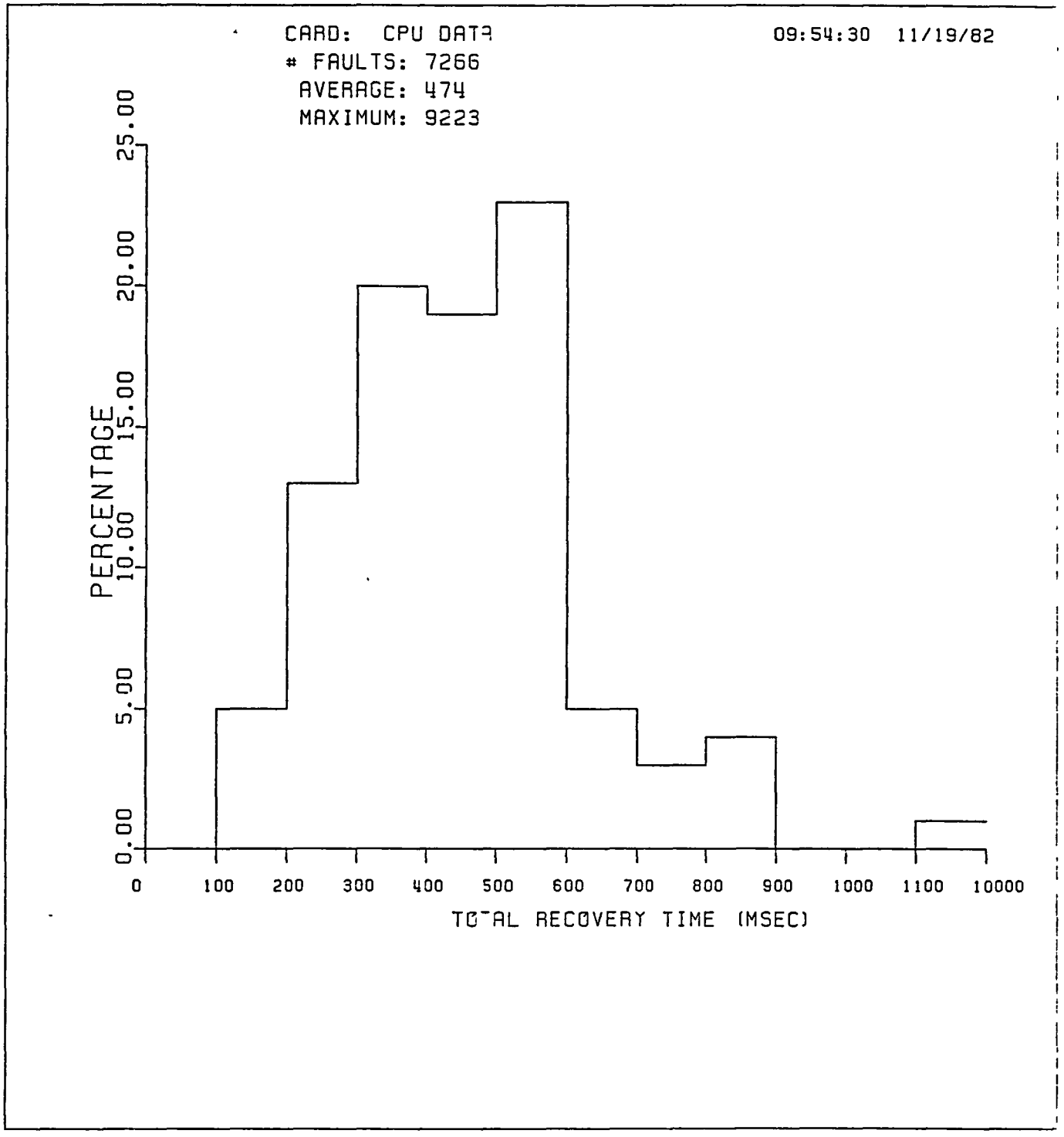
CARD: CPU DATA
FAULTS: 7266
AVERAGE: 474
MAXIMUM: 9223
09:54:30 11/19/82
PERCENTAGE
0.00
5.00
10.00
15.00
20.00
25.00
0
100
200
300
400
500
600
700
800
900
1000
1100
10000
TOTAL RECOVERY TIME (MSEC)

Figure 13

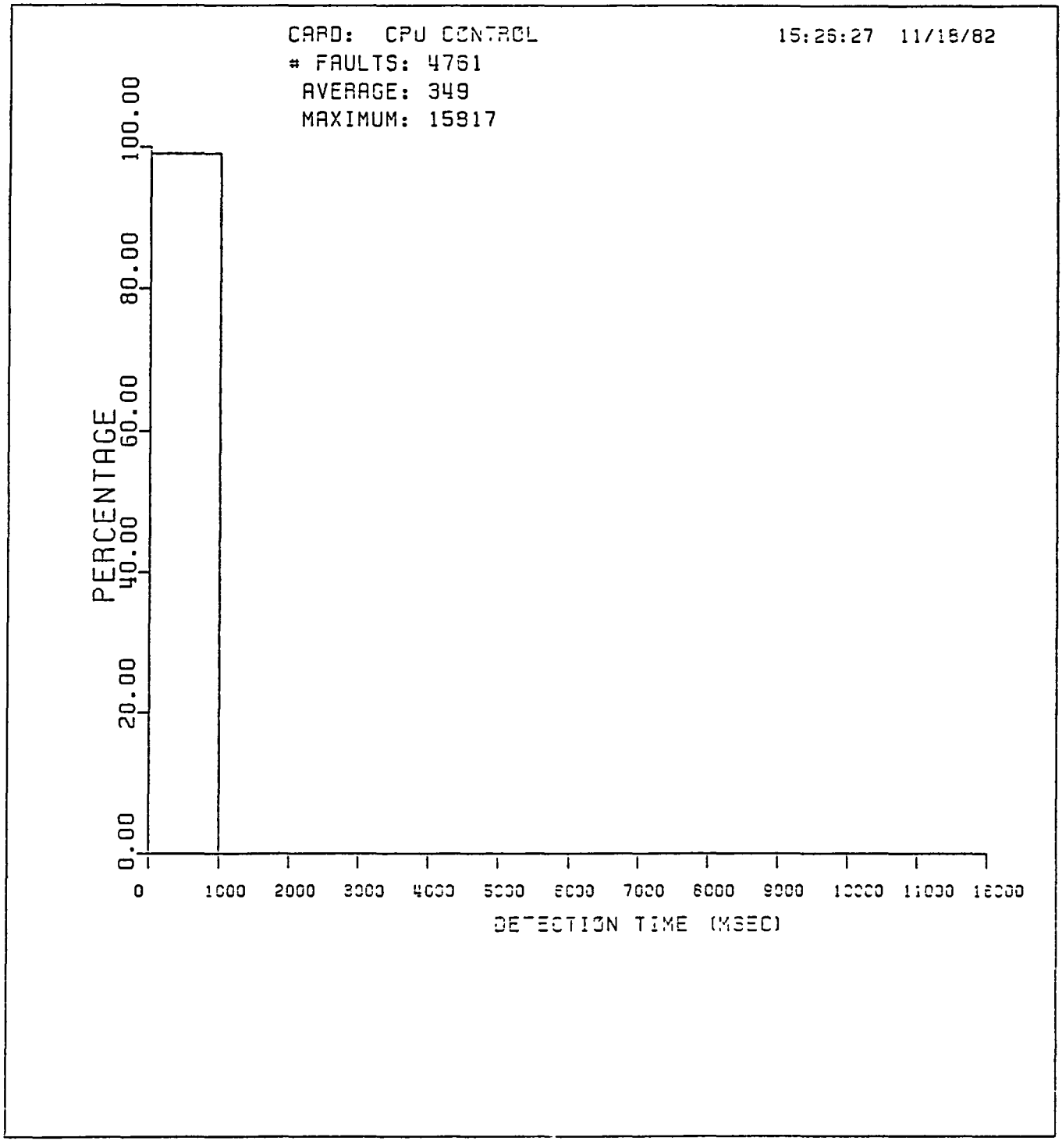
CARD: CPU CONTROL
15:26:27 11/18/82
FAULTS: 4761
AVERAGE: 349
MAXIMUM: 15817
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
DETECTION TIME (MSEC)

Figure 14

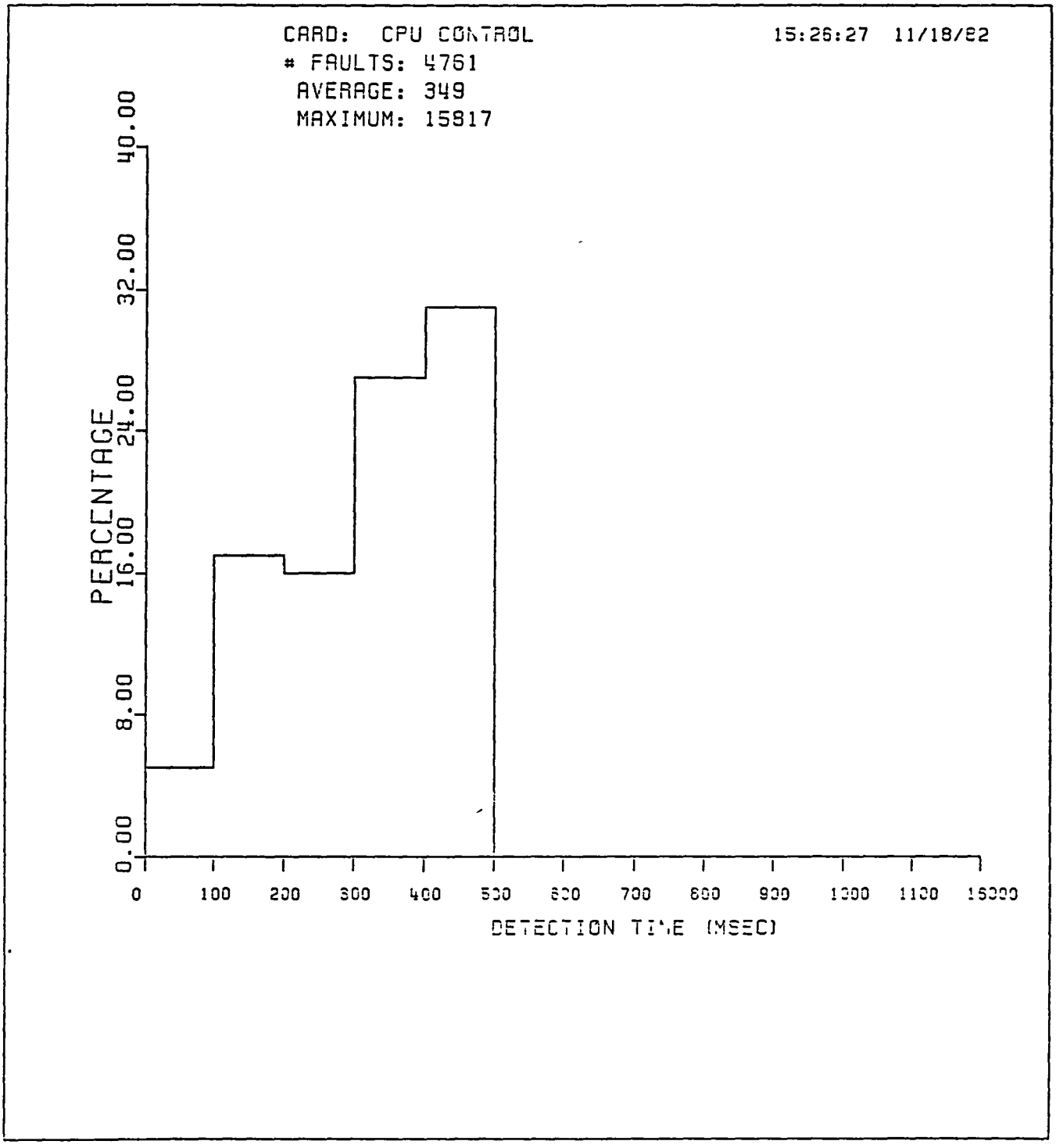
CARD: CPU CONTROL
15:26:27 11/18/82
FAULTS: 4761
AVERAGE: 349
MAXIMUM: 15817
PERCENTAGE
0.00
8.00
16.00
24.00
32.00
40.00
0
100
200
300
400
500
600
700
800
900
1000
1100
15000
DETECTION TIME (MSEC)

Figure 15

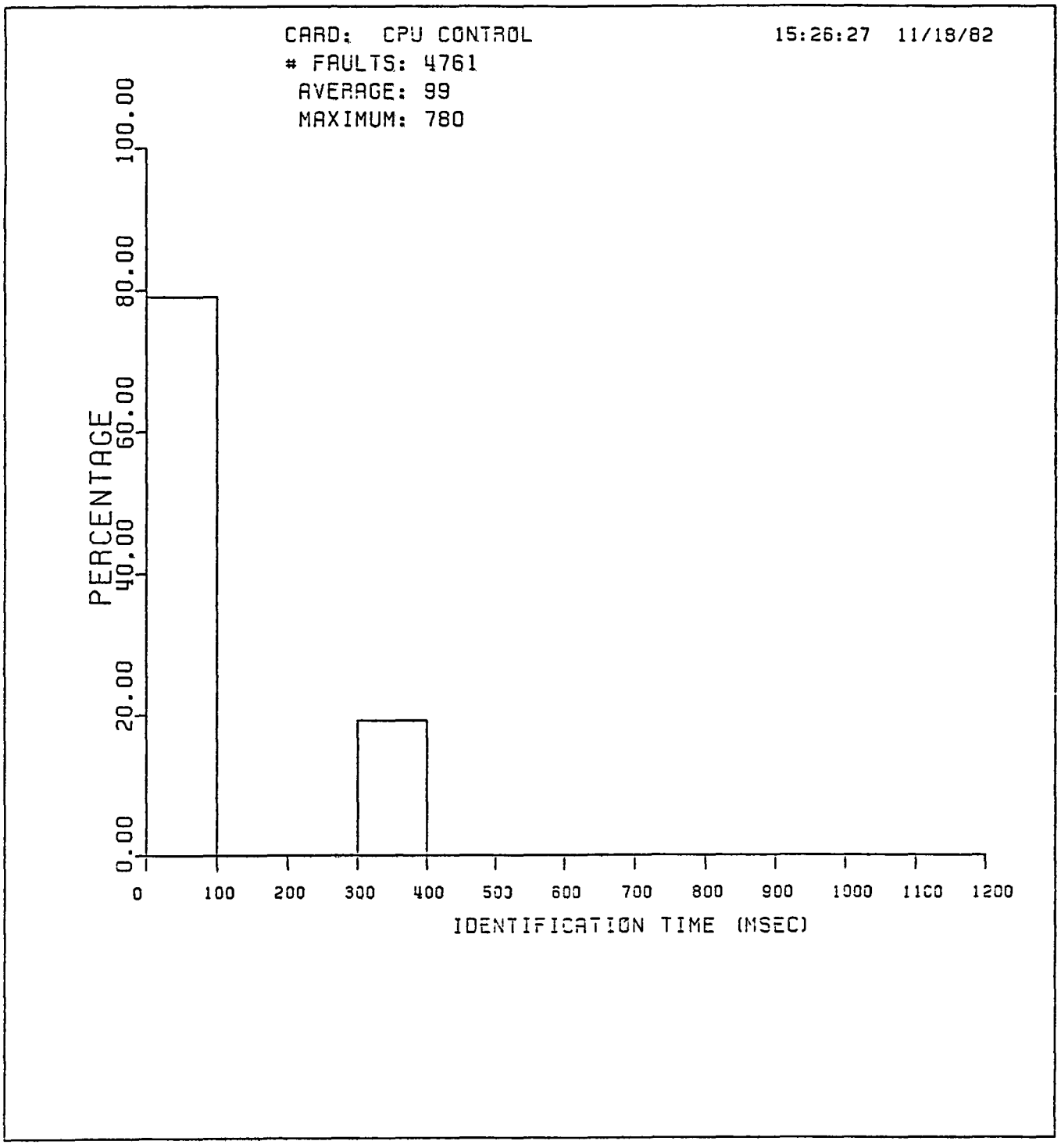
CARD: CPU CONTROL
15:26:27 11/18/82
FAULTS: 4761
AVERAGE: 99
MAXIMUM: 780
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 16

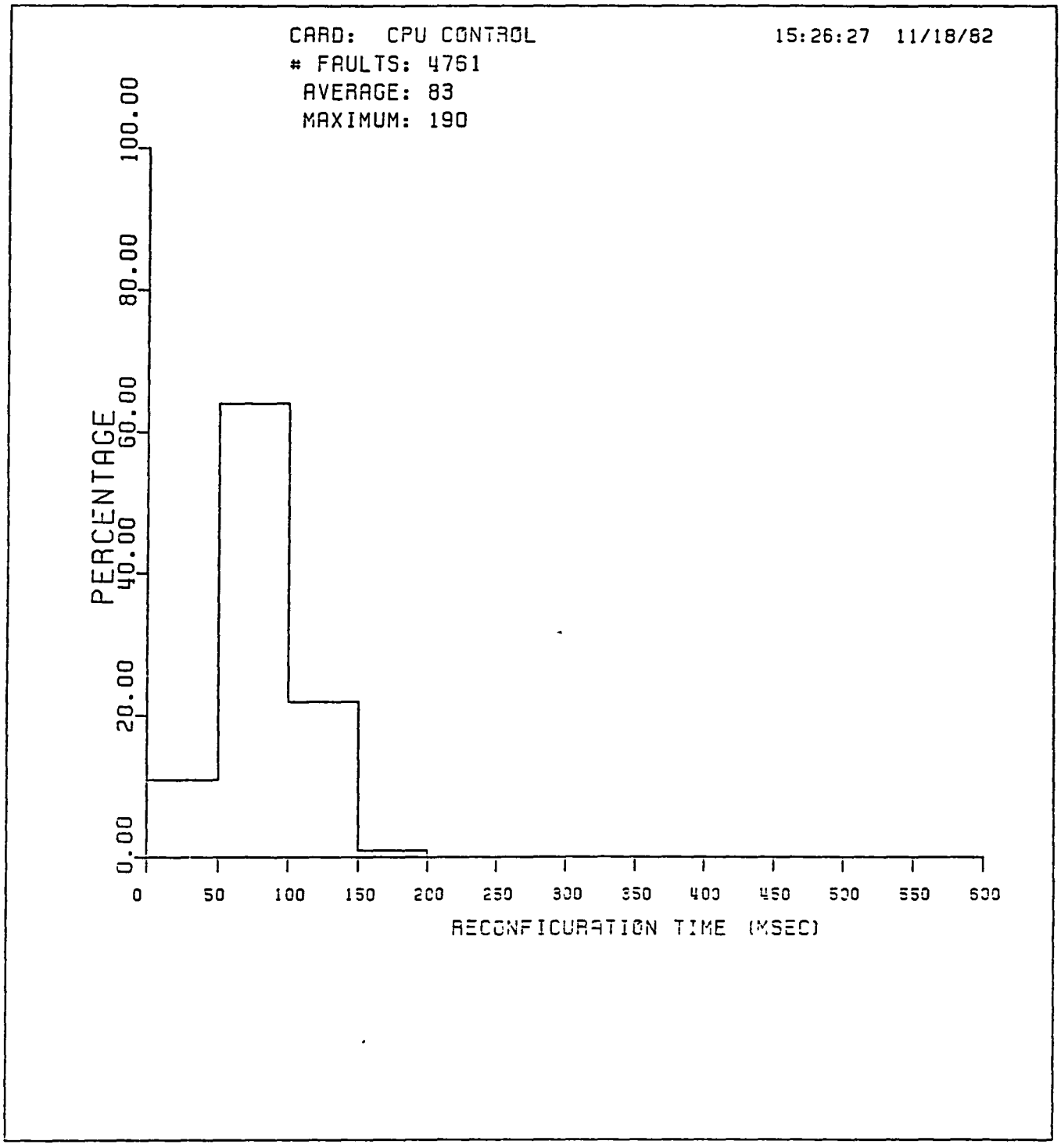
CARD: CPU CONTROL
15:26:27 11/18/82
FAULTS: 4761
AVERAGE: 83
MAXIMUM: 190
PERCENTAGE
0.00
20.00
40.00
60.00
80.00
100.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 17

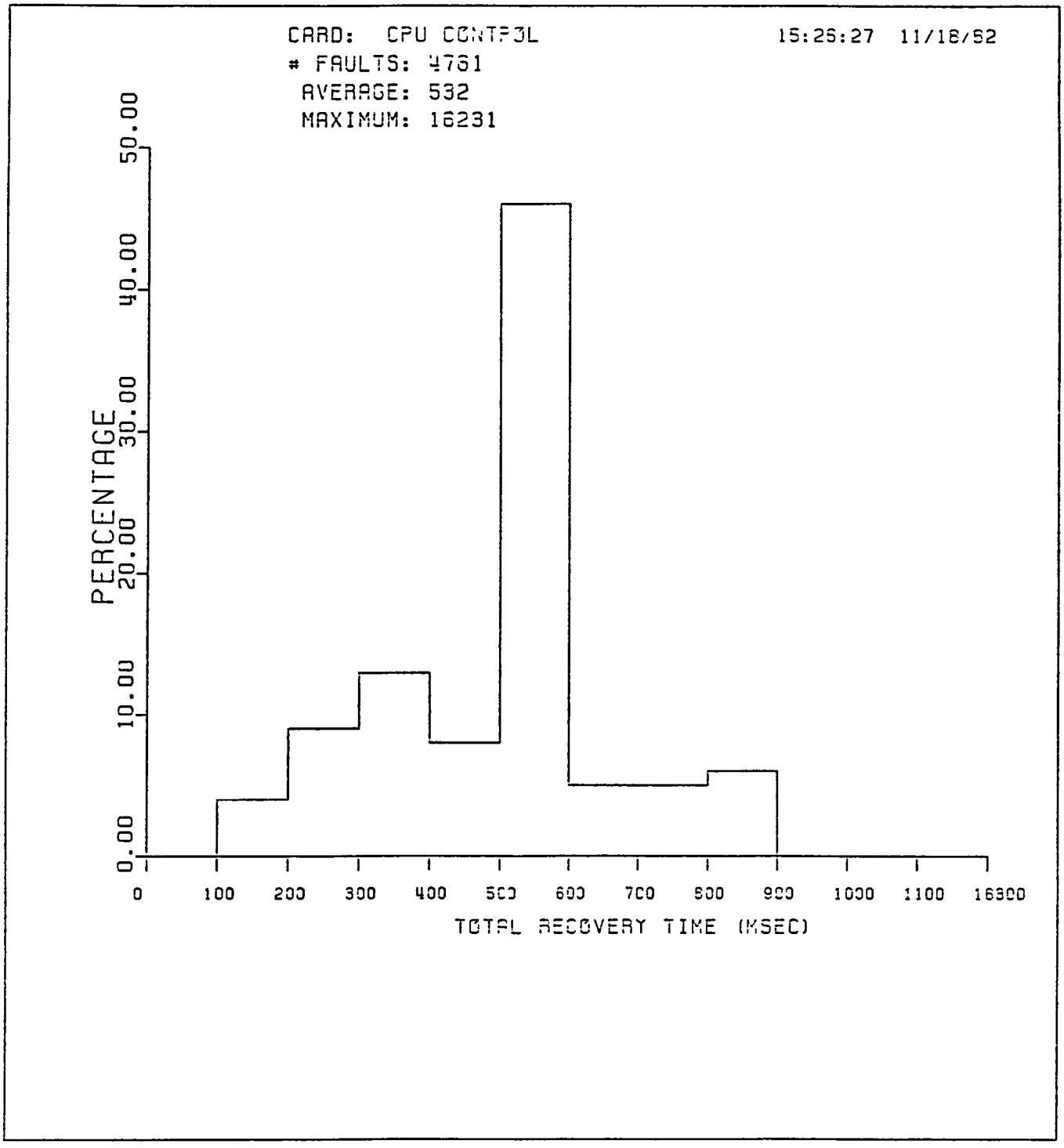
CARD: CPU CONTROL
FAULTS: 4761
AVERAGE: 532
MAXIMUM: 16231
15:25:27 11/18/82
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
100
200
300
400
500
600
700
800
900
1000
1100
16300
TOTAL RECOVERY TIME (MSEC)

Figure 18

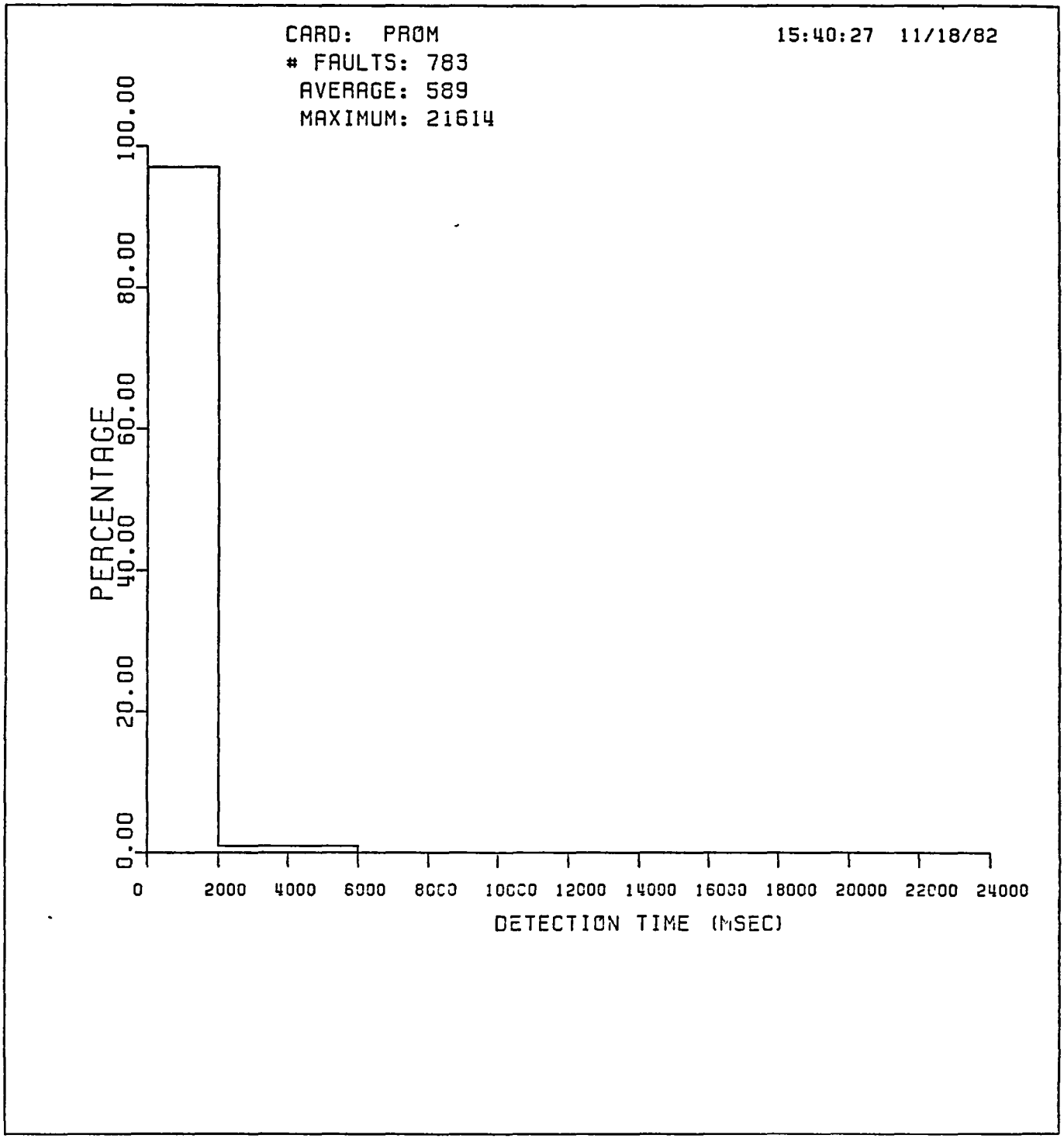
CARD: PROM
15:40:27 11/18/82
FAULTS: 783
AVERAGE: 589
MAXIMUM: 21614
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000
DETECTION TIME (MSEC)

Figure 19

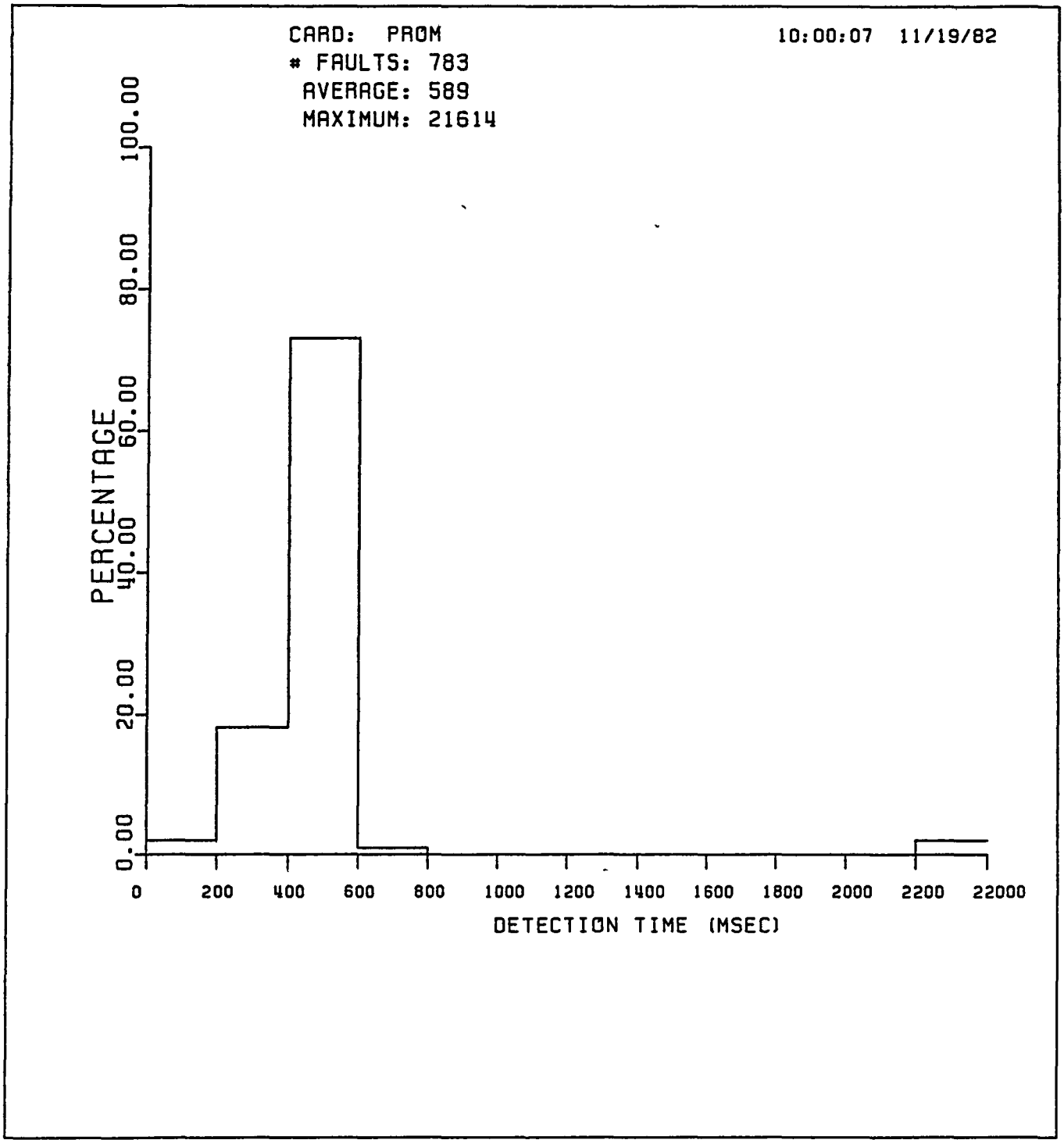
CARD: PROM
10:00:07 11/19/82
FAULTS: 783
AVERAGE: 589
MAXIMUM: 21614
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
22000
DETECTION TIME (MSEC)

Figure 20

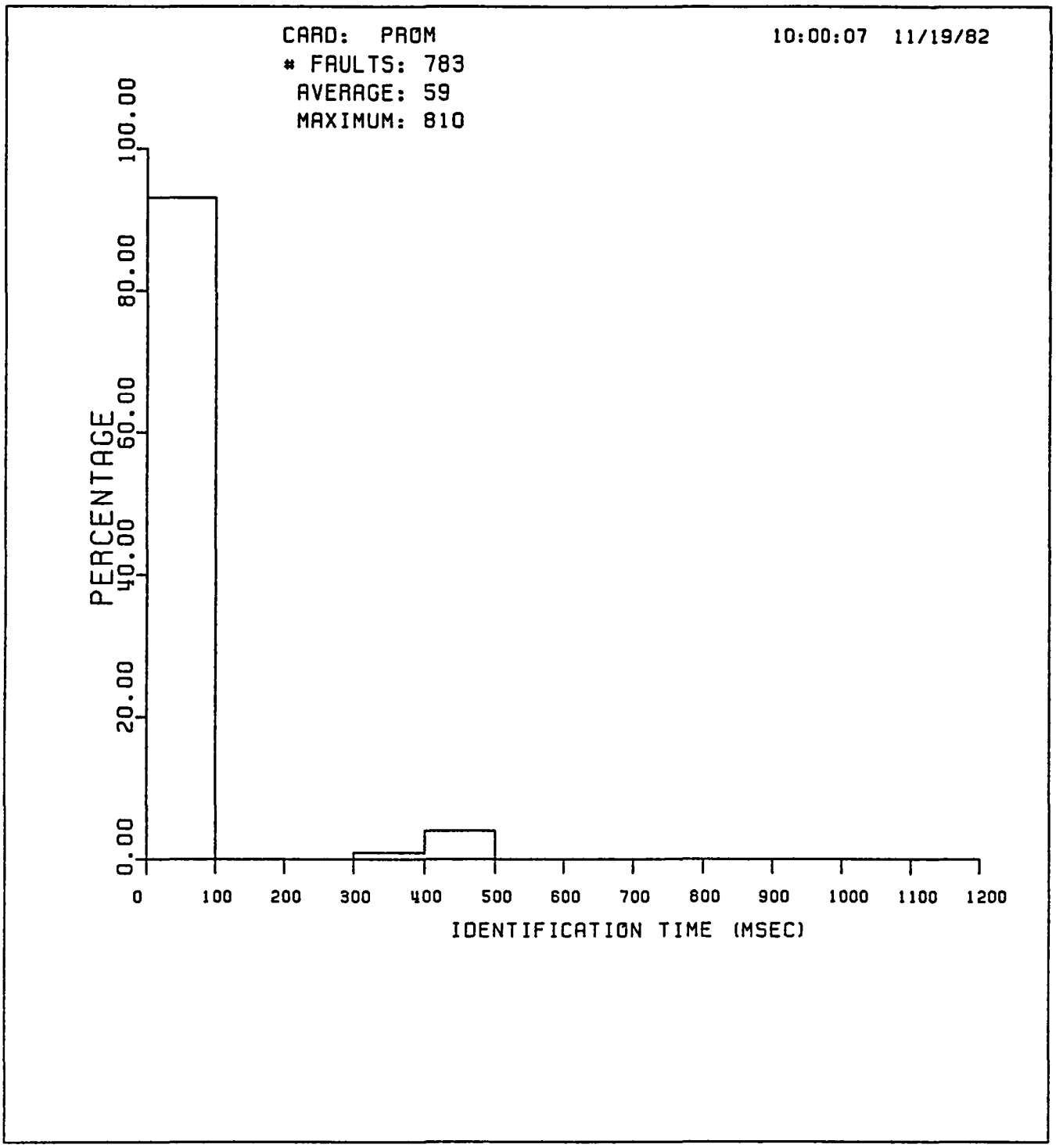
CARD: PROM
10:00:07 11/19/82
FAULTS: 783
AVERAGE: 59
MAXIMUM: 810
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 21

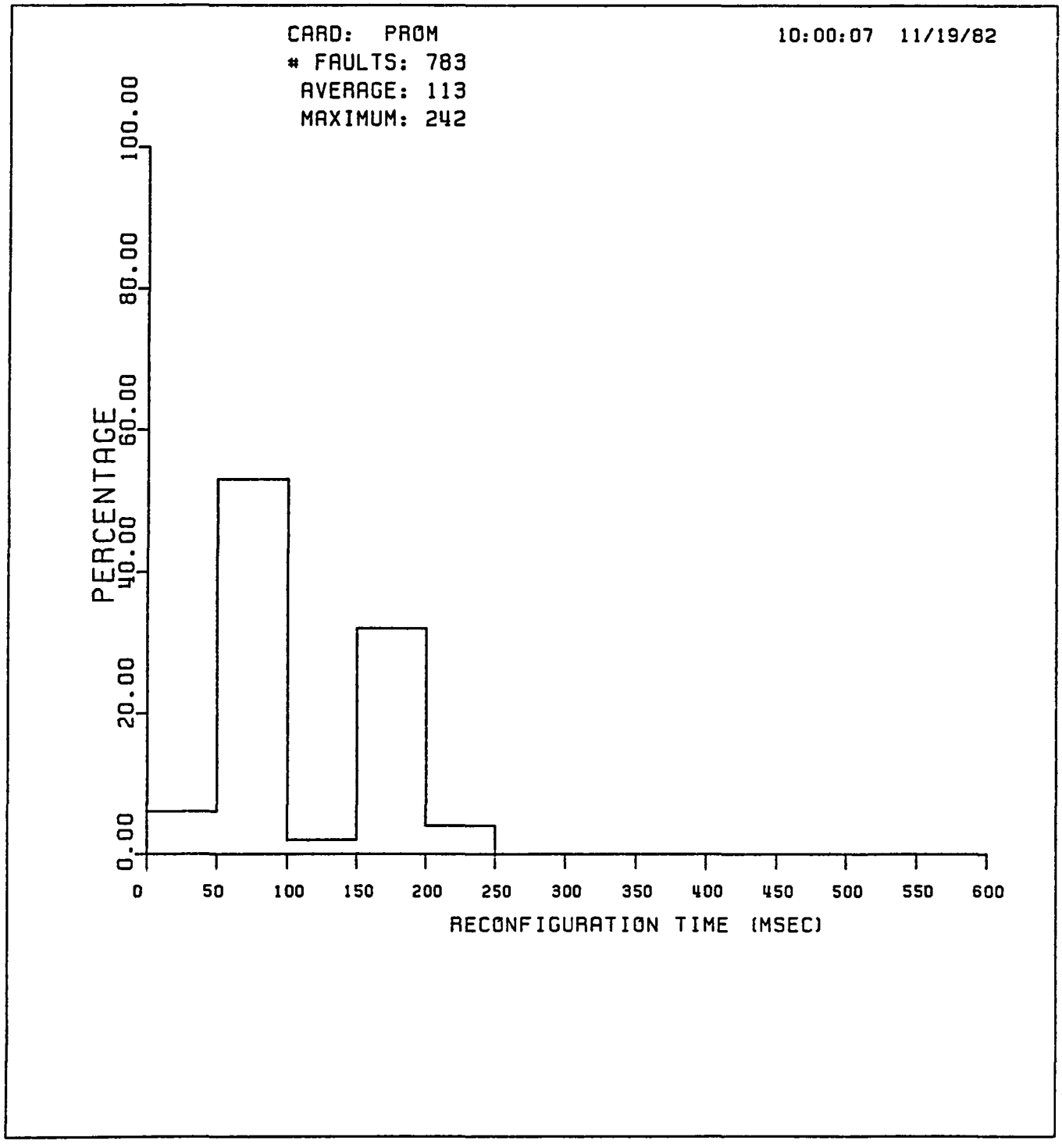
CARD: PROM
10:00:07 11/19/82
FAULTS: 783
AVERAGE: 113
MAXIMUM: 242
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 22

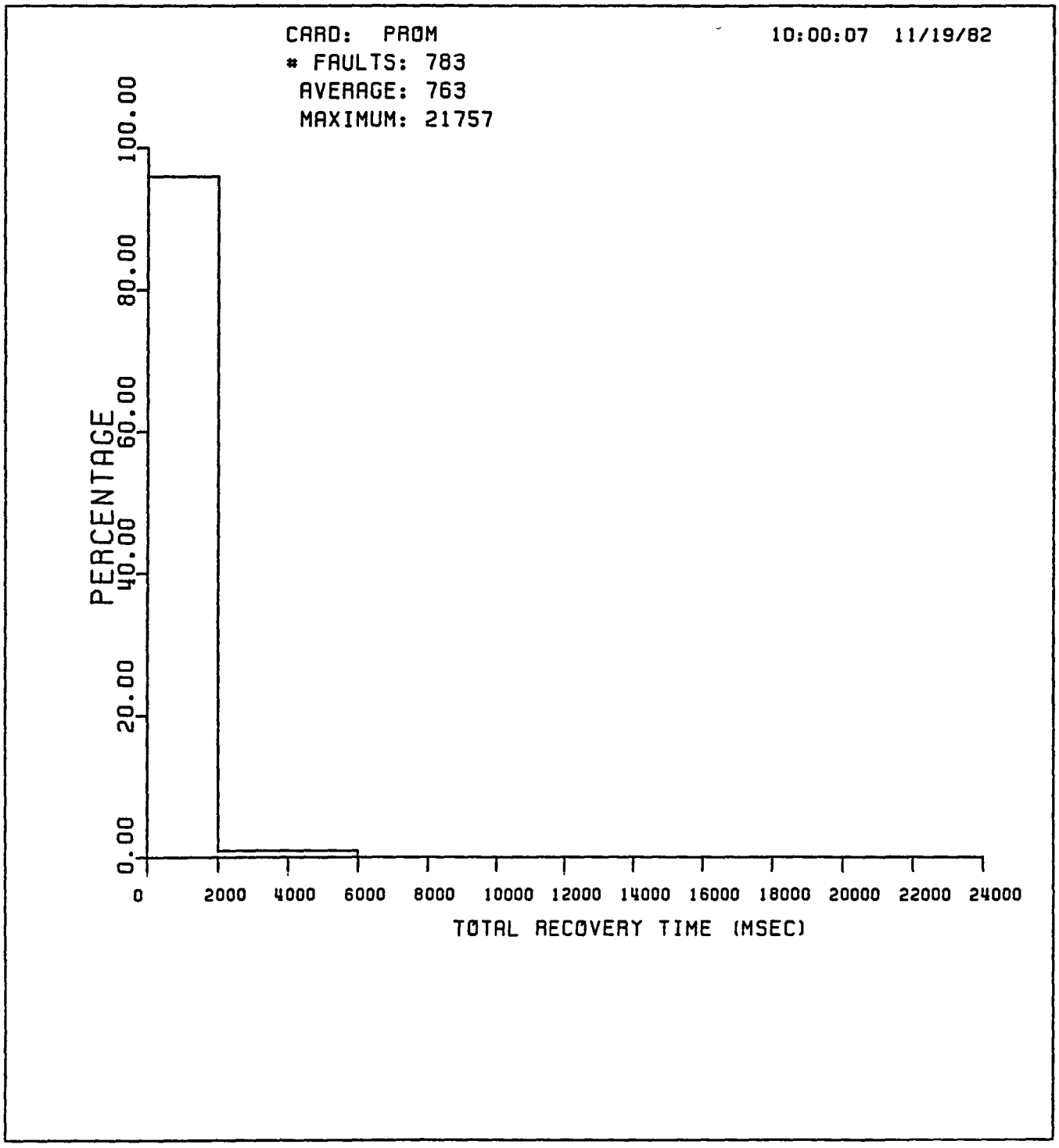
CARD: PROM
10:00:07 11/19/82
FAULTS: 783
AVERAGE: 763
MAXIMUM: 21757
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000
TOTAL RECOVERY TIME (MSEC)

Figure 23

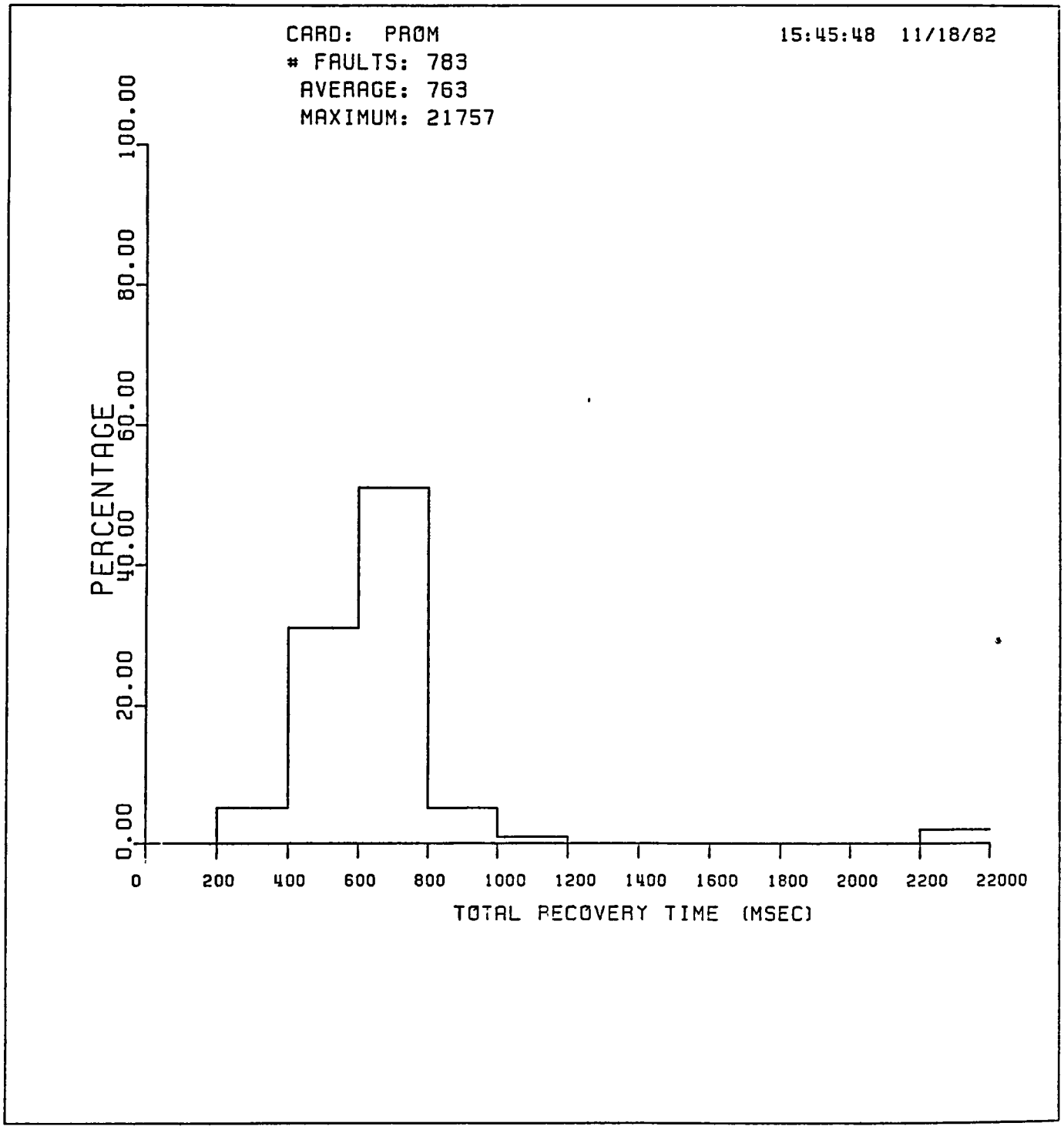
CARD: PROM
15:45:48 11/18/82
FAULTS: 783
AVERAGE: 763
MAXIMUM: 21757
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
22000
TOTAL RECOVERY TIME (MSEC)

Figure 24

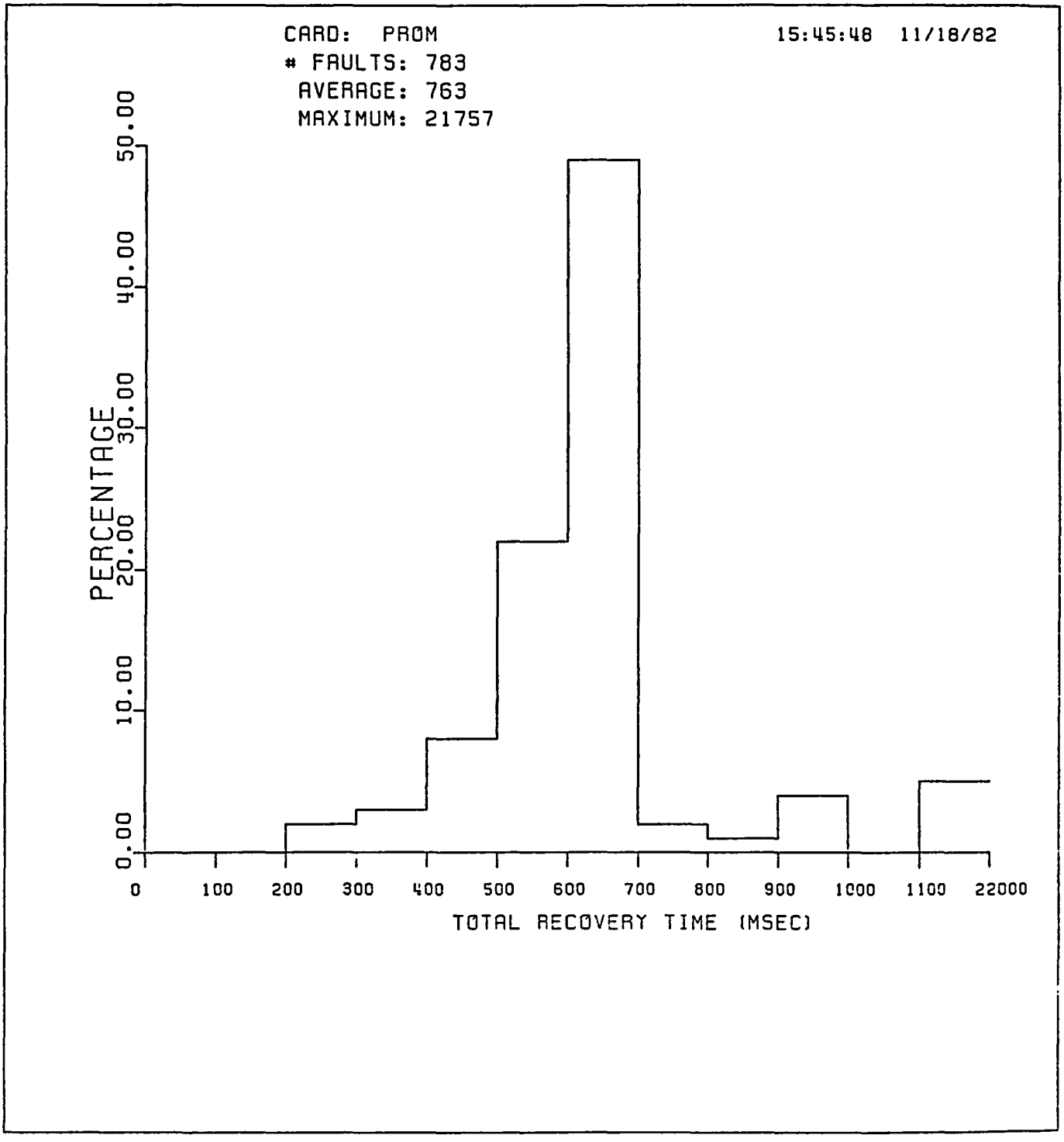
CARD: PROM
15:45:48 11/18/82
FAULTS: 783
AVERAGE: 763
MAXIMUM: 21757
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
100
200
300
400
500
600
700
800
900
1000
1100
22000
TOTAL RECOVERY TIME (MSEC)

Figure 25

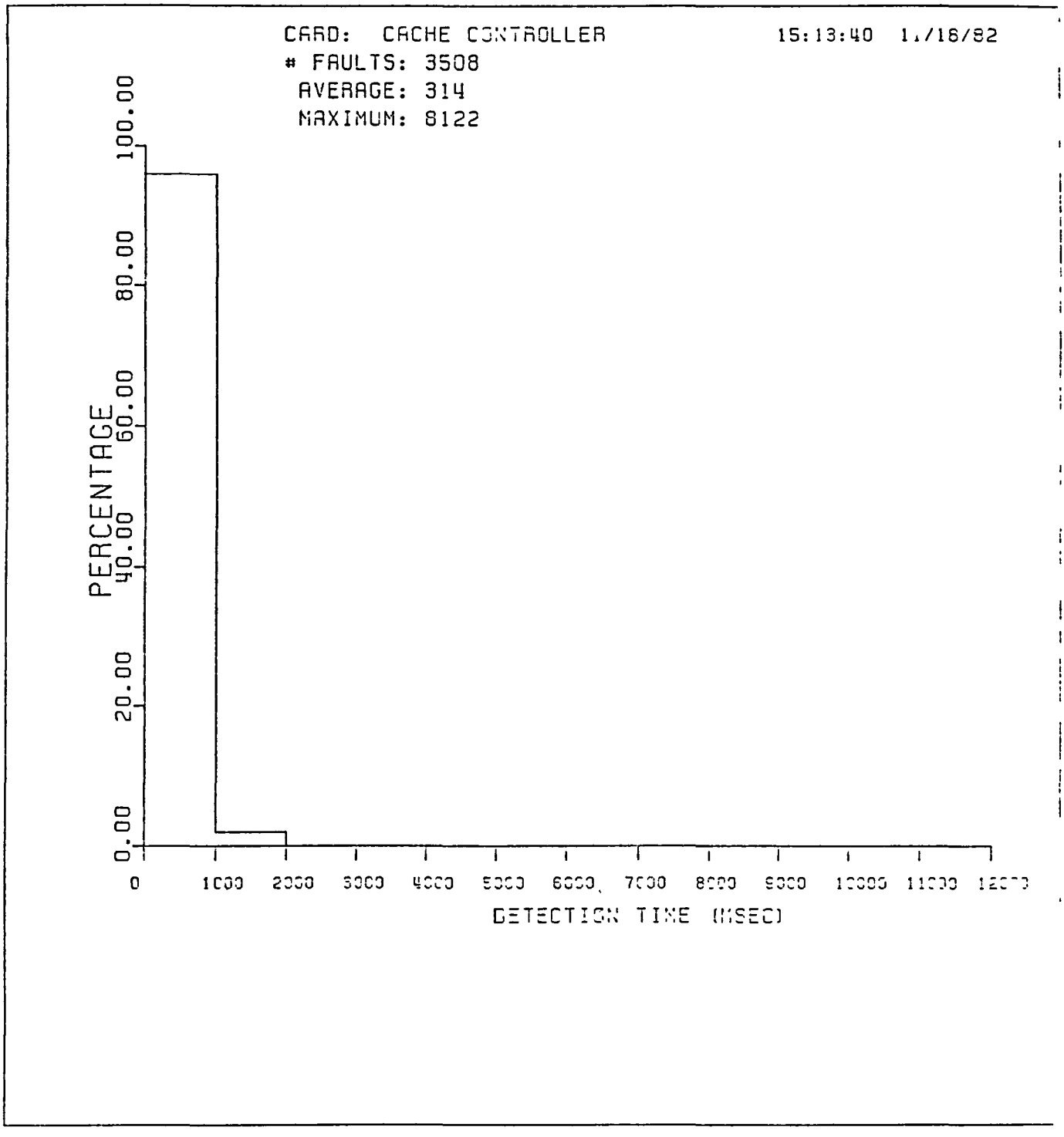
CARD: CACHE CONTROLLER
15:13:40 1./18/82
FAULTS: 3508
AVERAGE: 314
MAXIMUM: 8122
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
DETECTION TIME (MSEC)

Figure 26

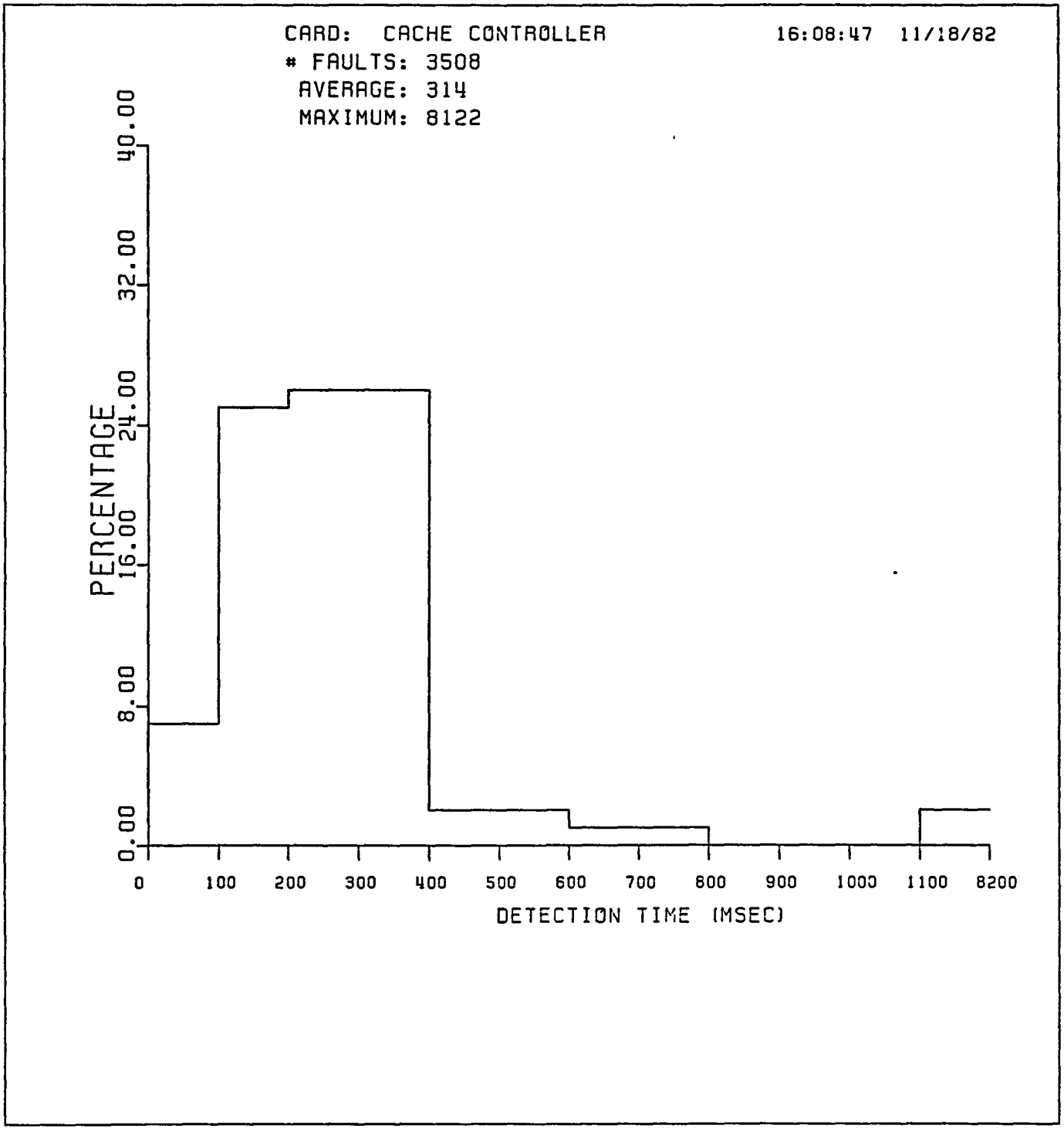
CARD: CACHE CONTROLLER
16:08:47 11/18/82
FAULTS: 3508
AVERAGE: 314
MAXIMUM: 8122
PERCENTAGE
40.00
32.00
24.00
16.00
8.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
8200
DETECTION TIME (MSEC)

Figure 27

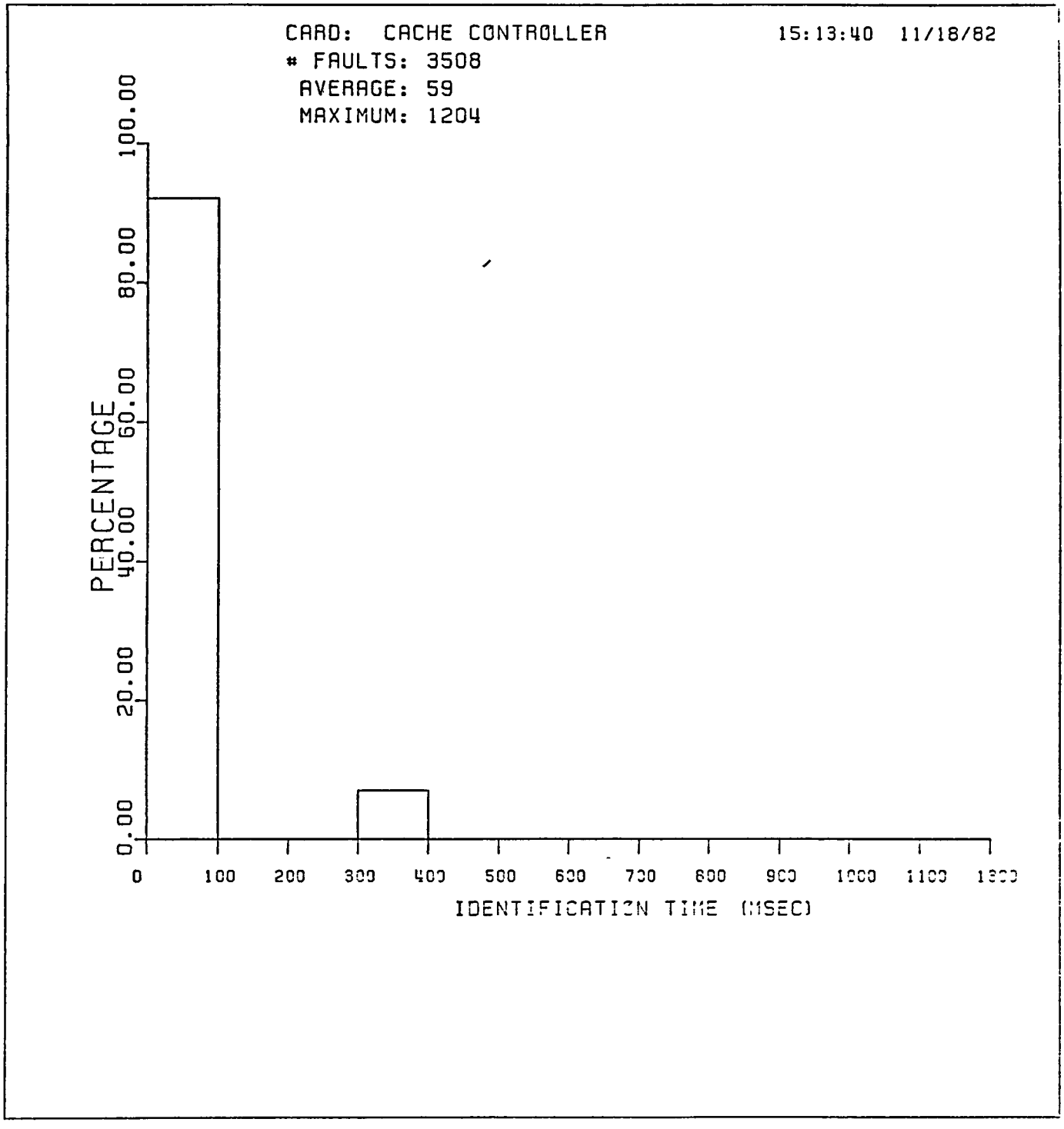
CARD: CACHE CONTROLLER
15:13:40 11/18/82
FAULTS: 3508
AVERAGE: 59
MAXIMUM: 1204
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 28

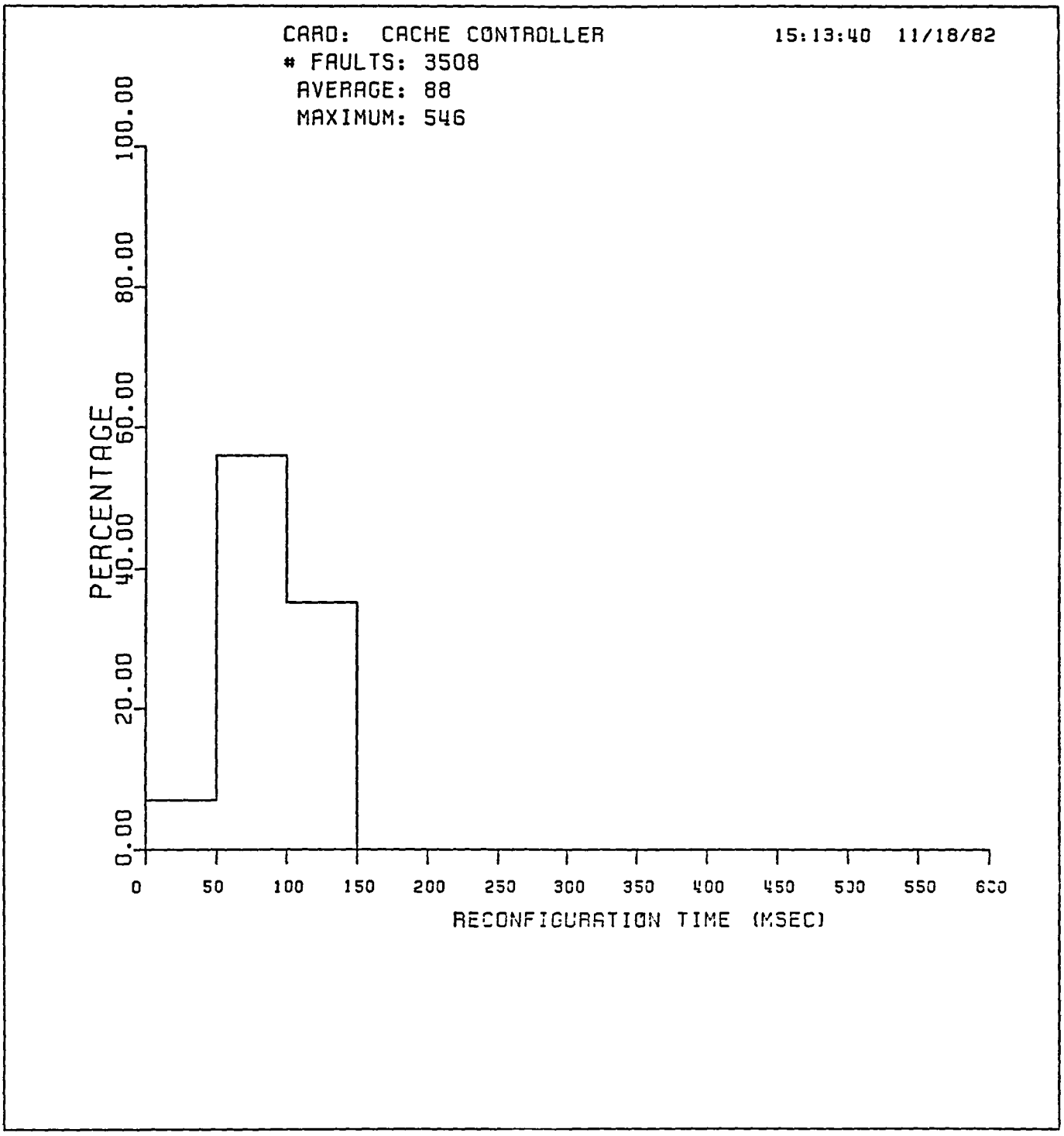
CARD: CACHE CONTROLLER
15:13:40 11/18/82
FAULTS: 3508
AVERAGE: 88
MAXIMUM: 546
PERCENTAGE
0.00
20.00
40.00
60.00
80.00
100.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 29

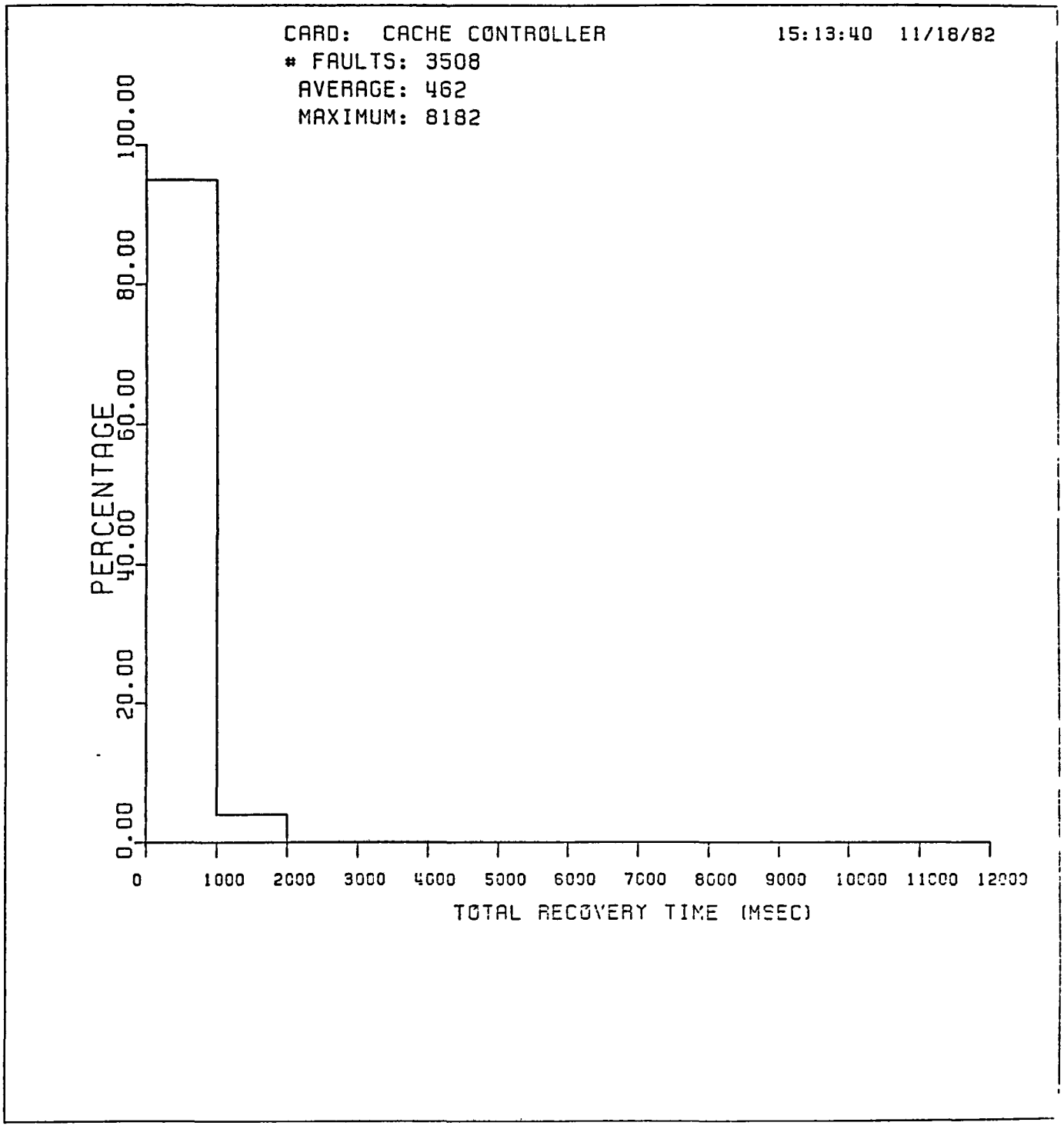
CARD: CACHE CONTROLLER
15:13:40 11/18/82
FAULTS: 3508
AVERAGE: 462
MAXIMUM: 8182
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
TOTAL RECOVERY TIME (MSEC)

Figure 30

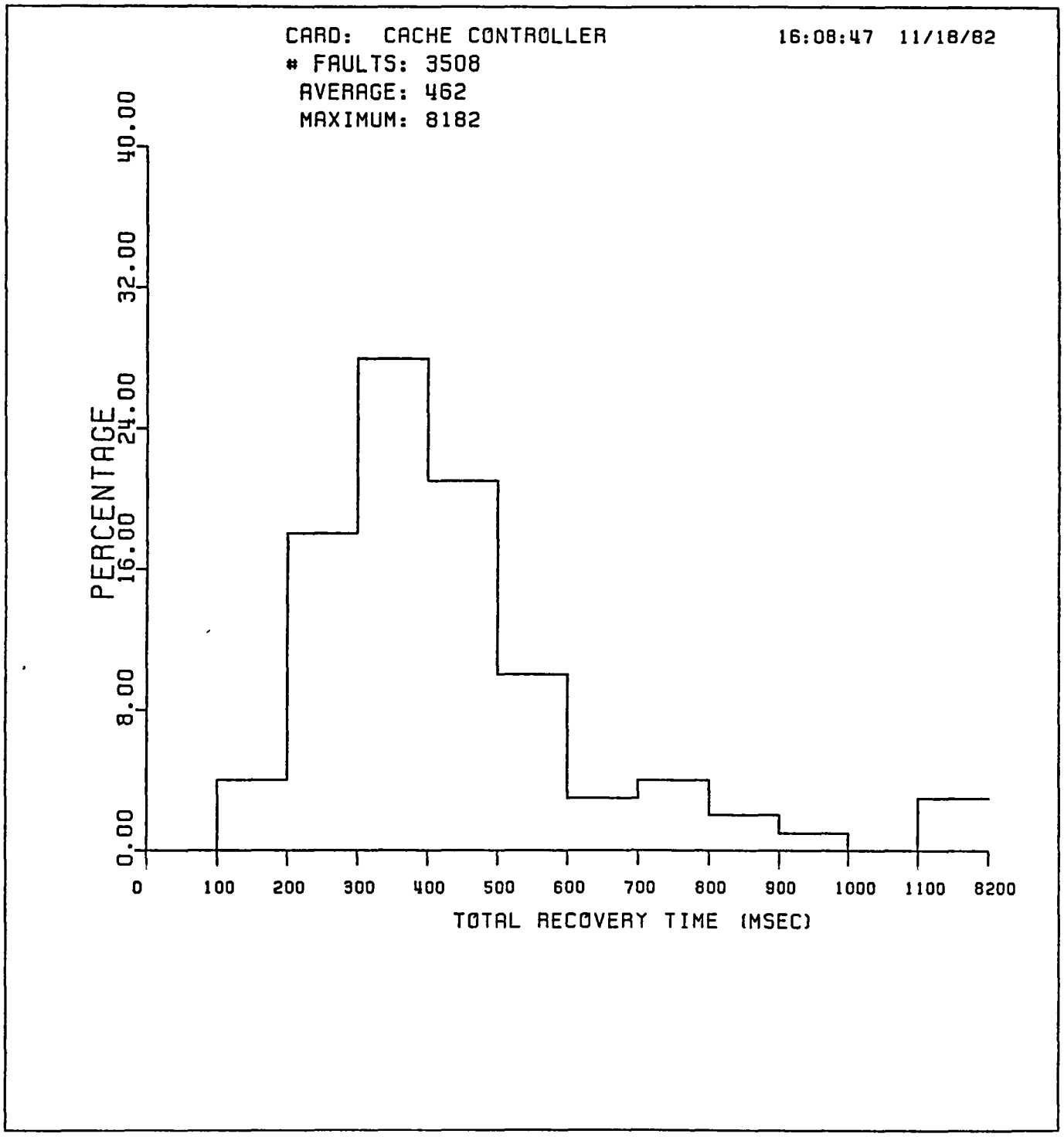
CARD: CACHE CONTROLLER
16:08:47 11/18/82
FAULTS: 3508
AVERAGE: 462
MAXIMUM: 8182
PERCENTAGE
0.00
8.00
16.00
24.00
32.00
40.00
0
100
200
300
400
500
600
700
800
900
1000
1100
8200
TOTAL RECOVERY TIME (MSEC)

Figure 31

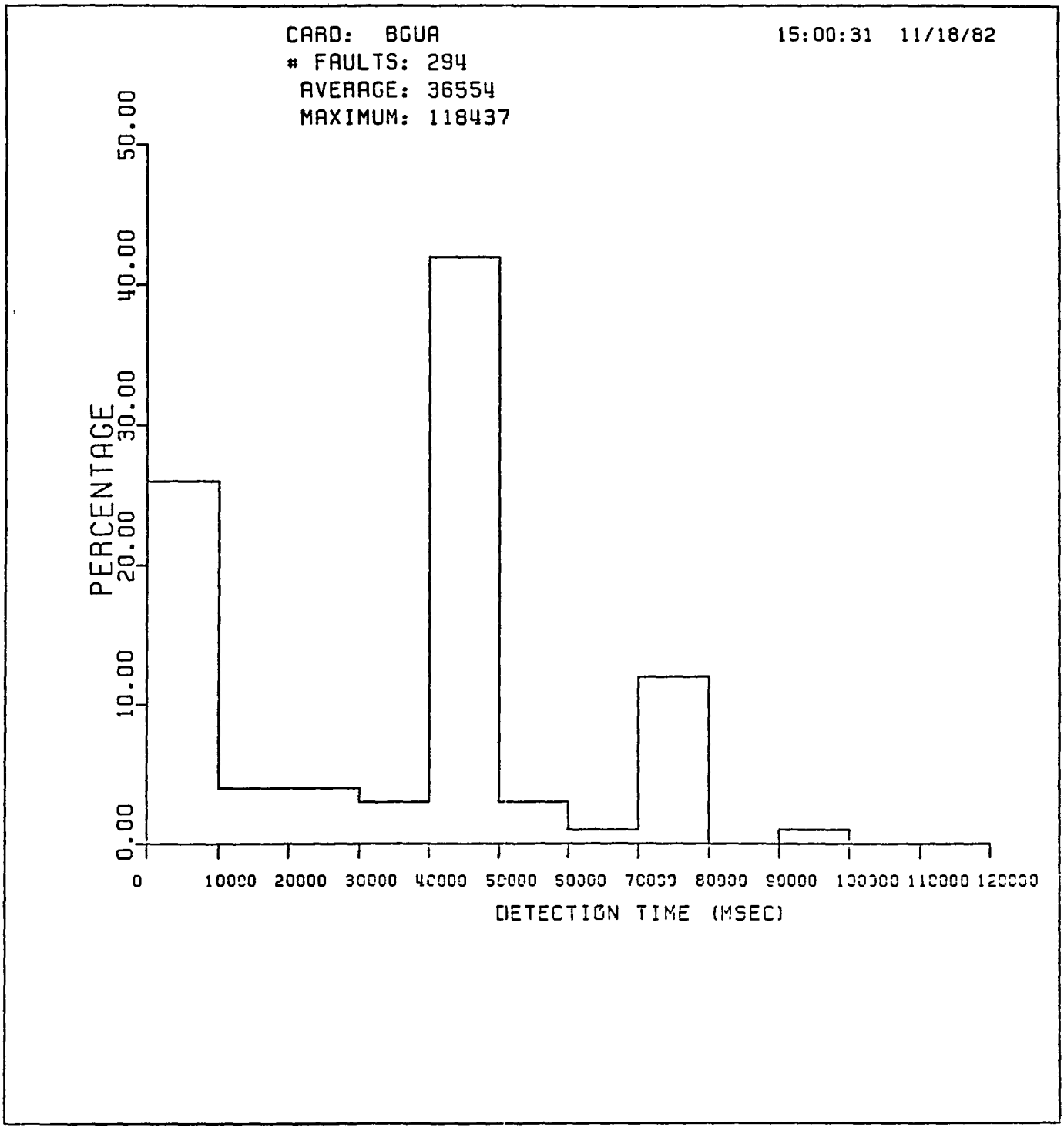
CARD: BGUA
15:00:31 11/18/82
FAULTS: 294
AVERAGE: 36554
MAXIMUM: 118437
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
DETECTION TIME (MSEC)

Figure 32

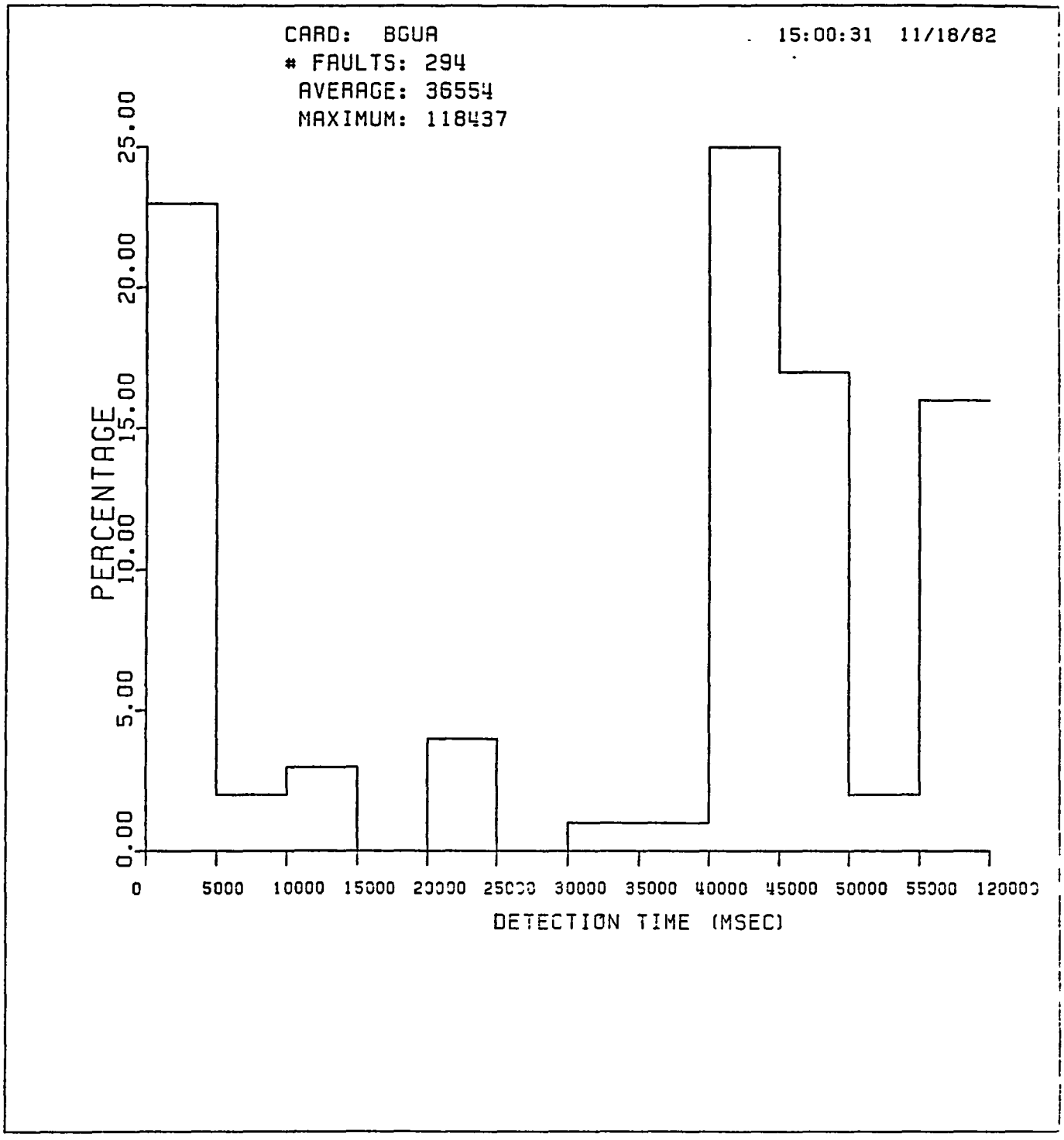
CARD: BGUA
15:00:31 11/18/82
FAULTS: 294
AVERAGE: 36554
MAXIMUM: 118437
PERCENTAGE
0.00
5.00
10.00
15.00
20.00
25.00
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
55000
120000
DETECTION TIME (MSEC)

Figure 33

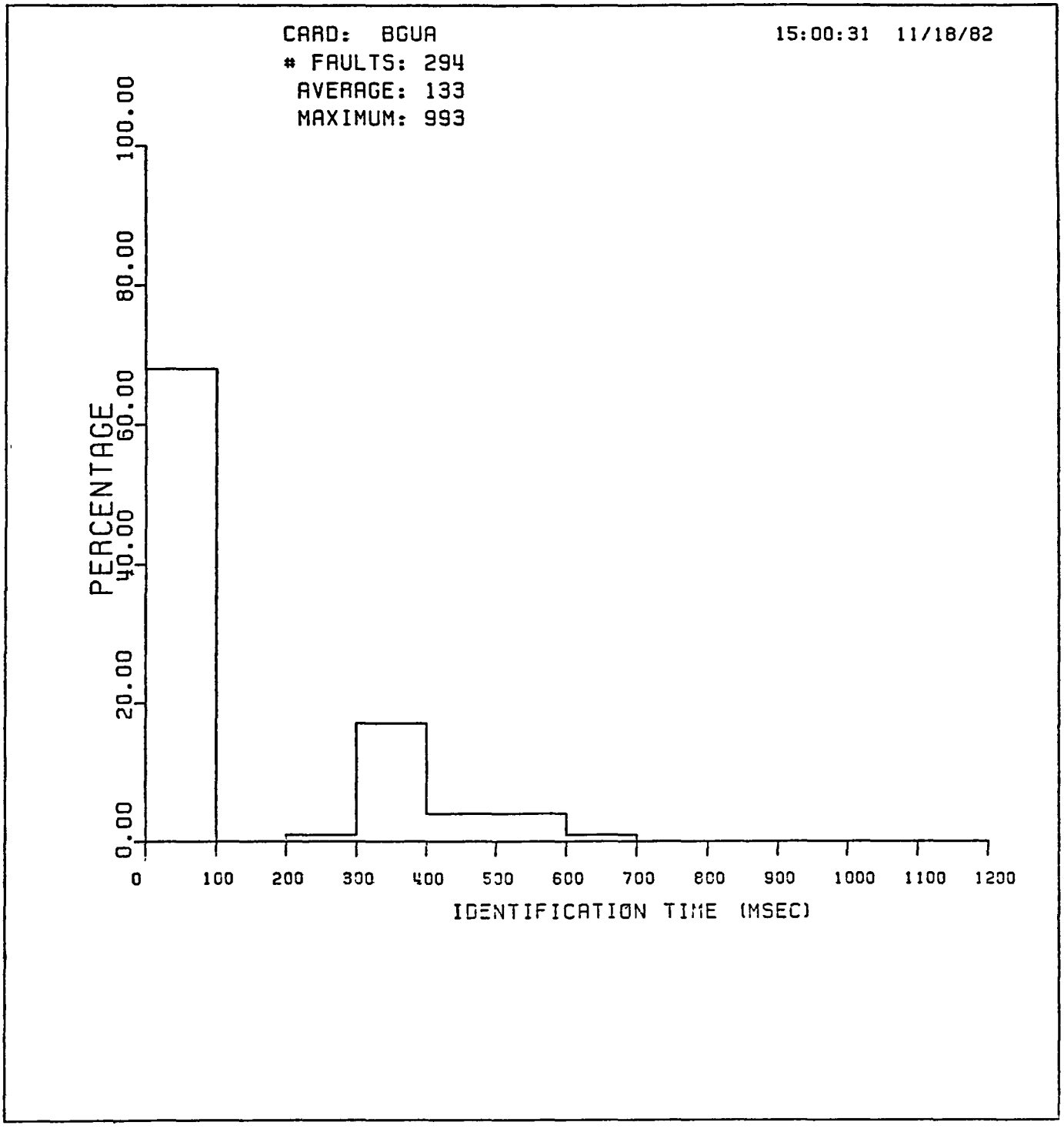
CARD: BGUA
15:00:31 11/18/82
FAULTS: 294
AVERAGE: 133
MAXIMUM: 993
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 34

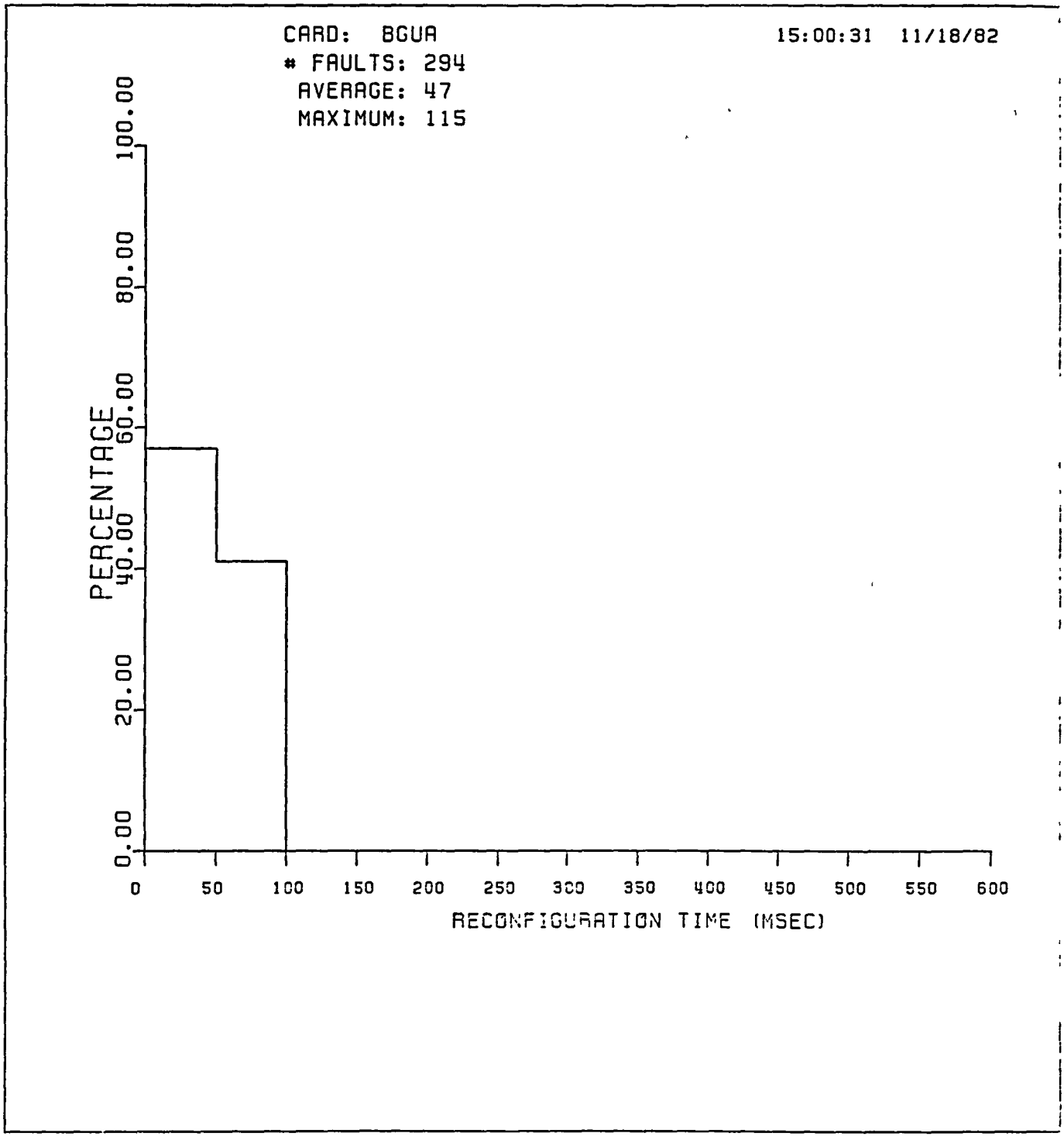
CARD: BGUA
15:00:31 11/18/82
FAULTS: 294
AVERAGE: 47
MAXIMUM: 115
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 35

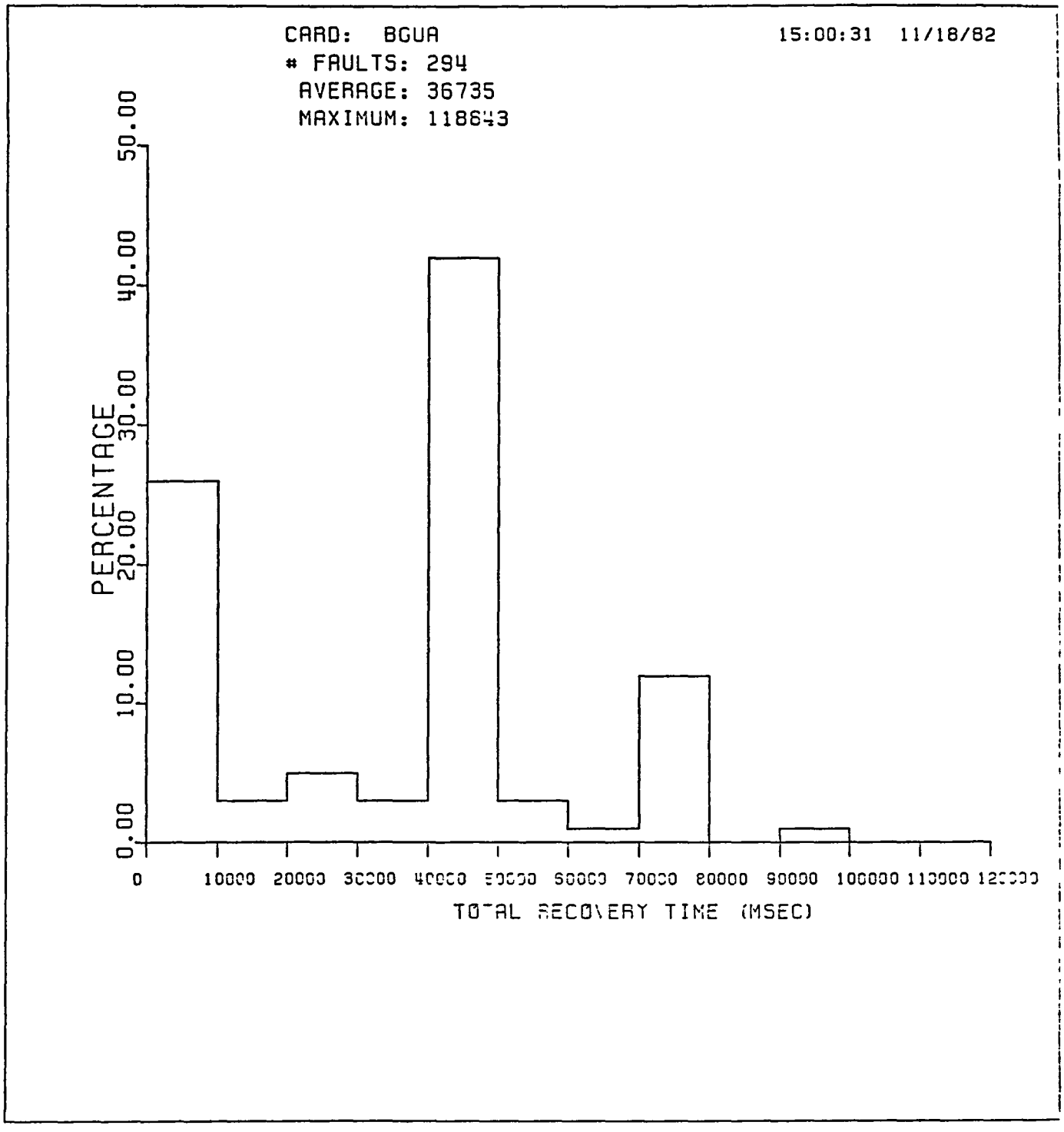
CARD: BGUA
15:00:31 11/18/82
FAULTS: 294
AVERAGE: 36735
MAXIMUM: 118643
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
TOTAL RECOVERY TIME (MSEC)

Figure 36

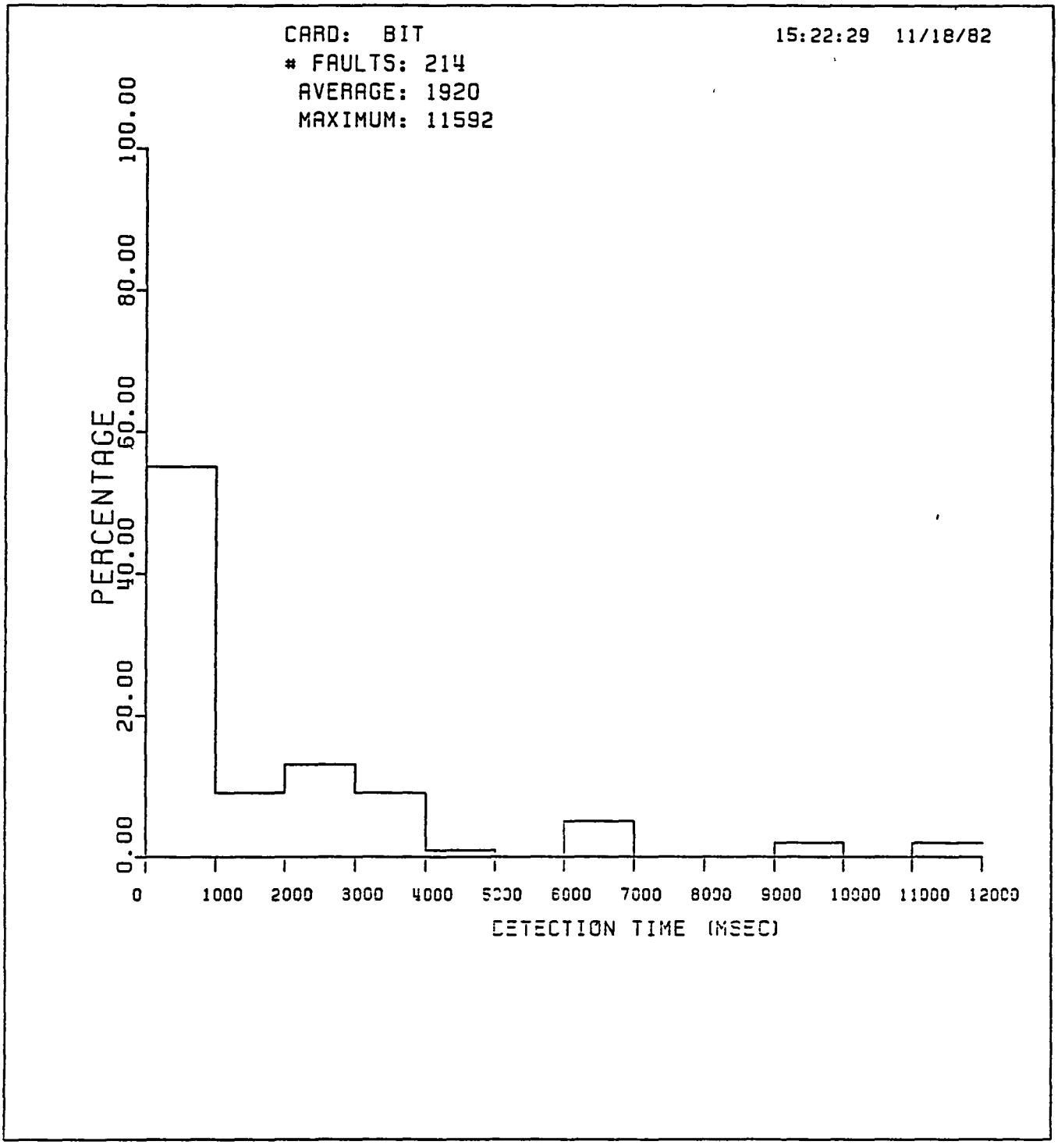
CARD: BIT
15:22:29 11/18/82
FAULTS: 214
AVERAGE: 1920
MAXIMUM: 11592
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
DETECTION TIME (MSEC)

Figure 37

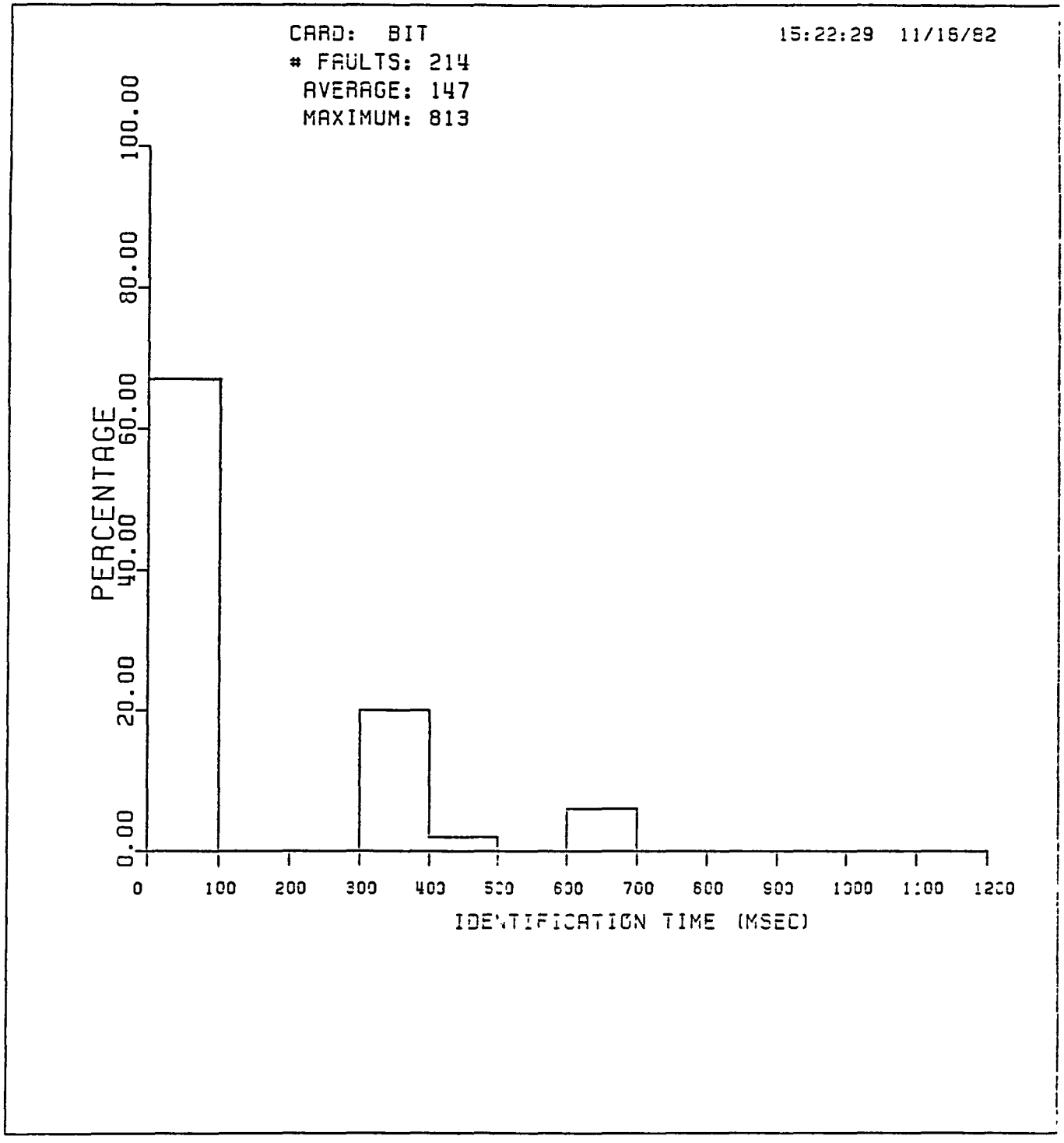
CARD: BIT
15:22:29 11/16/82
FAULTS: 214
AVERAGE: 147
MAXIMUM: 813
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 38

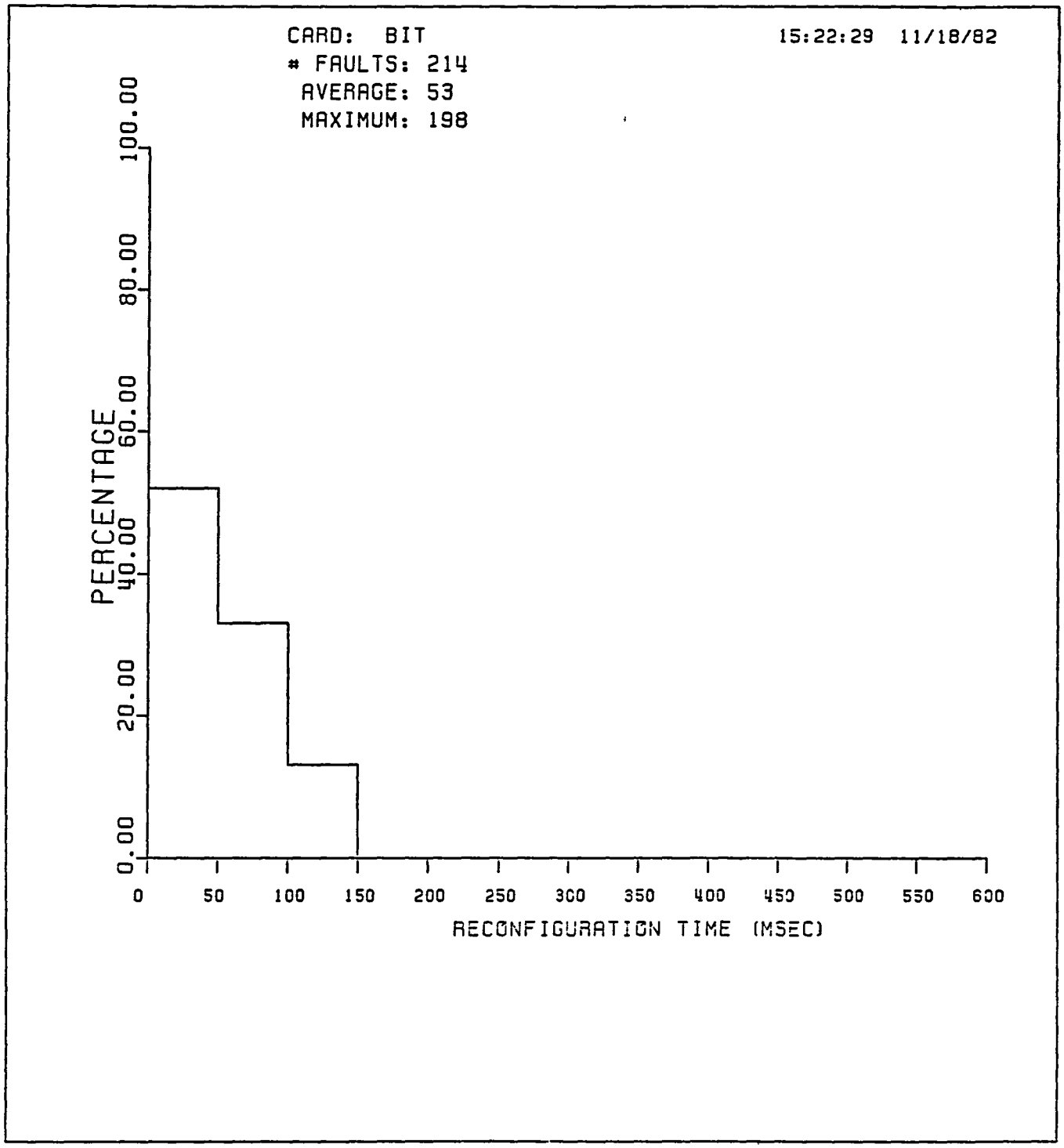
CARD: BIT
15:22:29 11/18/82
FAULTS: 214
AVERAGE: 53
MAXIMUM: 198
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 39

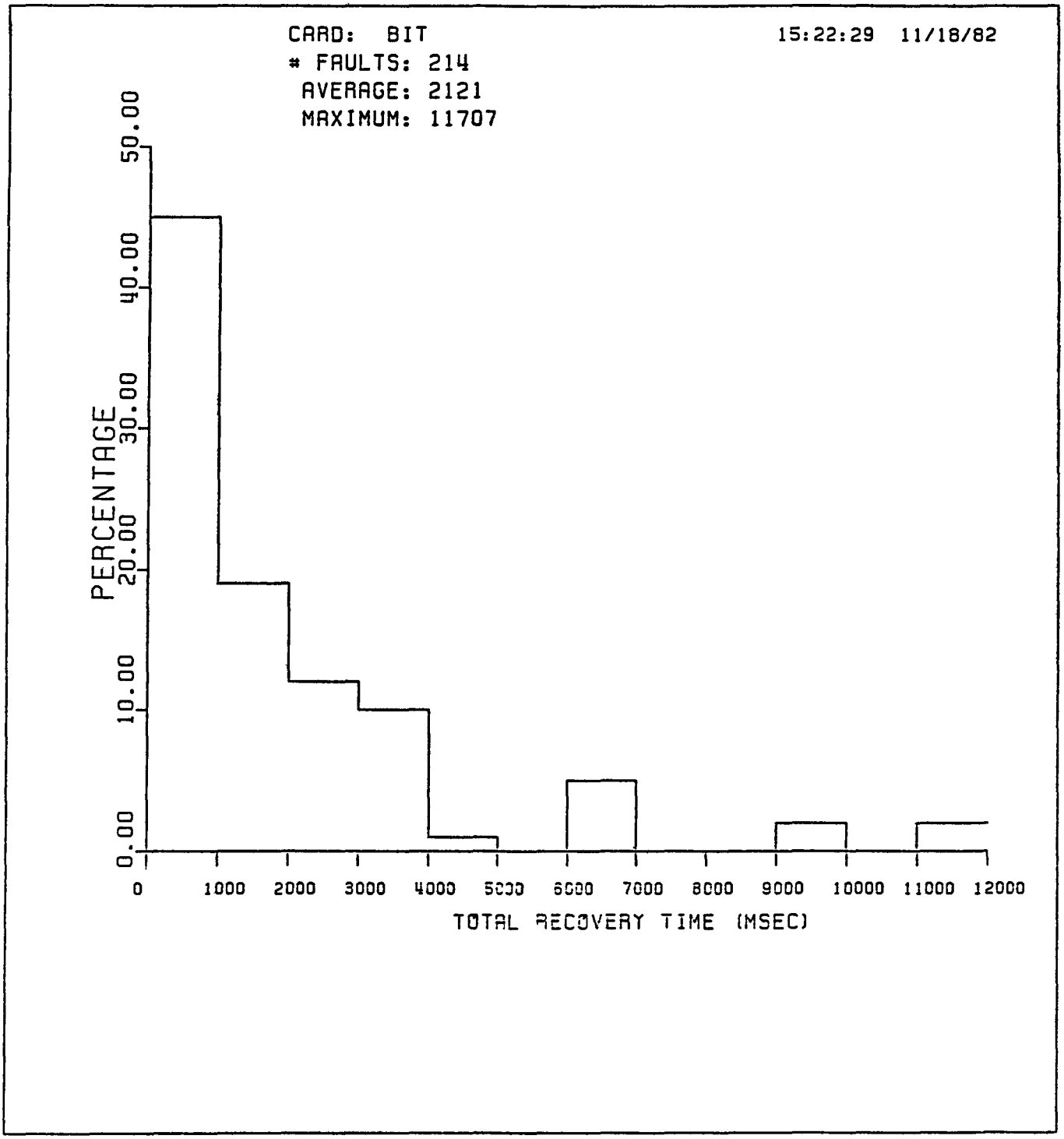
CARD: BIT
15:22:29 11/18/82
FAULTS: 214
AVERAGE: 2121
MAXIMUM: 11707
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
TOTAL RECOVERY TIME (MSEC)

Figure 40

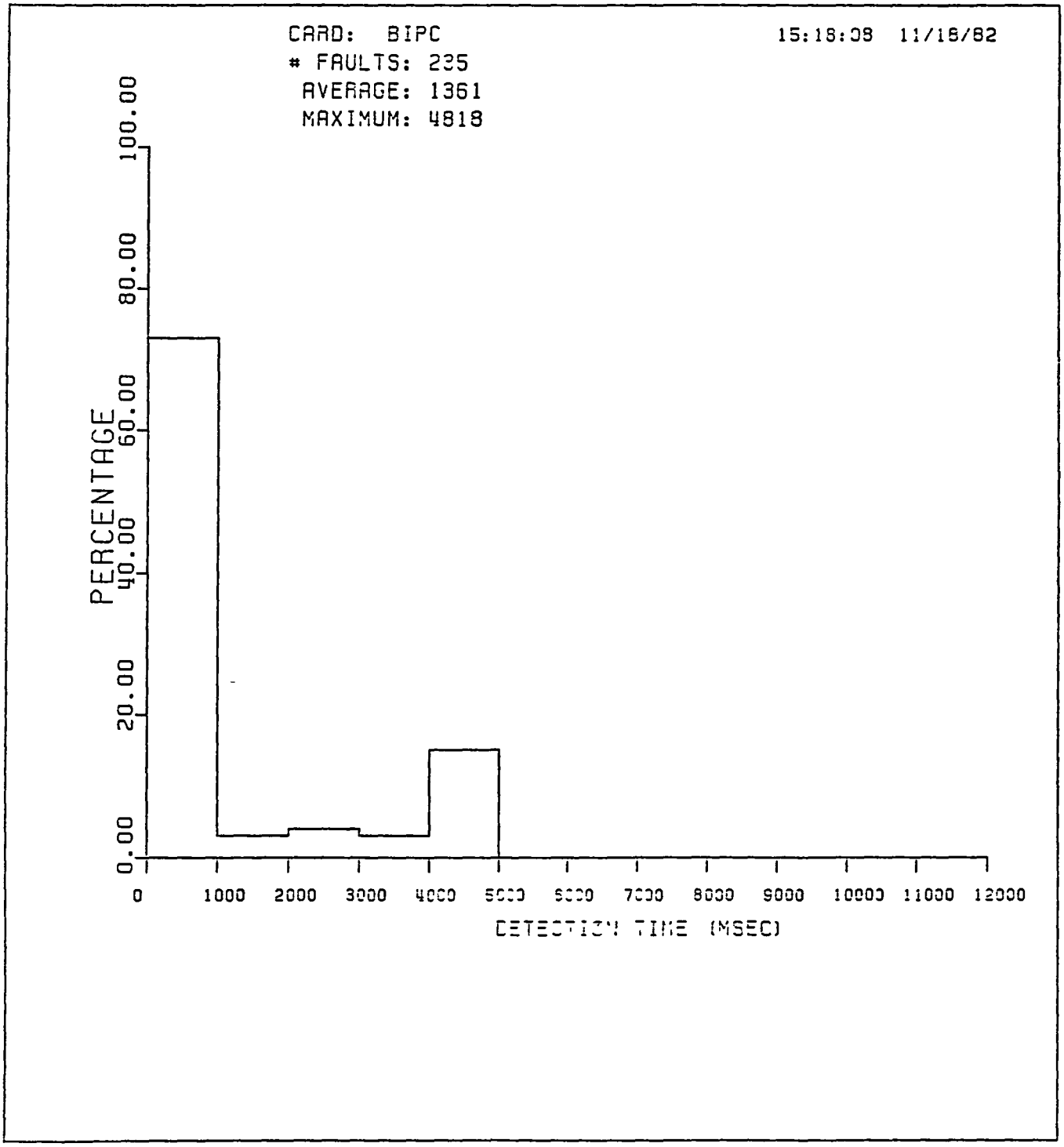
CARD: BIPC
15:18:08 11/18/82
FAULTS: 235
AVERAGE: 1361
MAXIMUM: 4818
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
DETECTION TIME (MSEC)

Figure 41

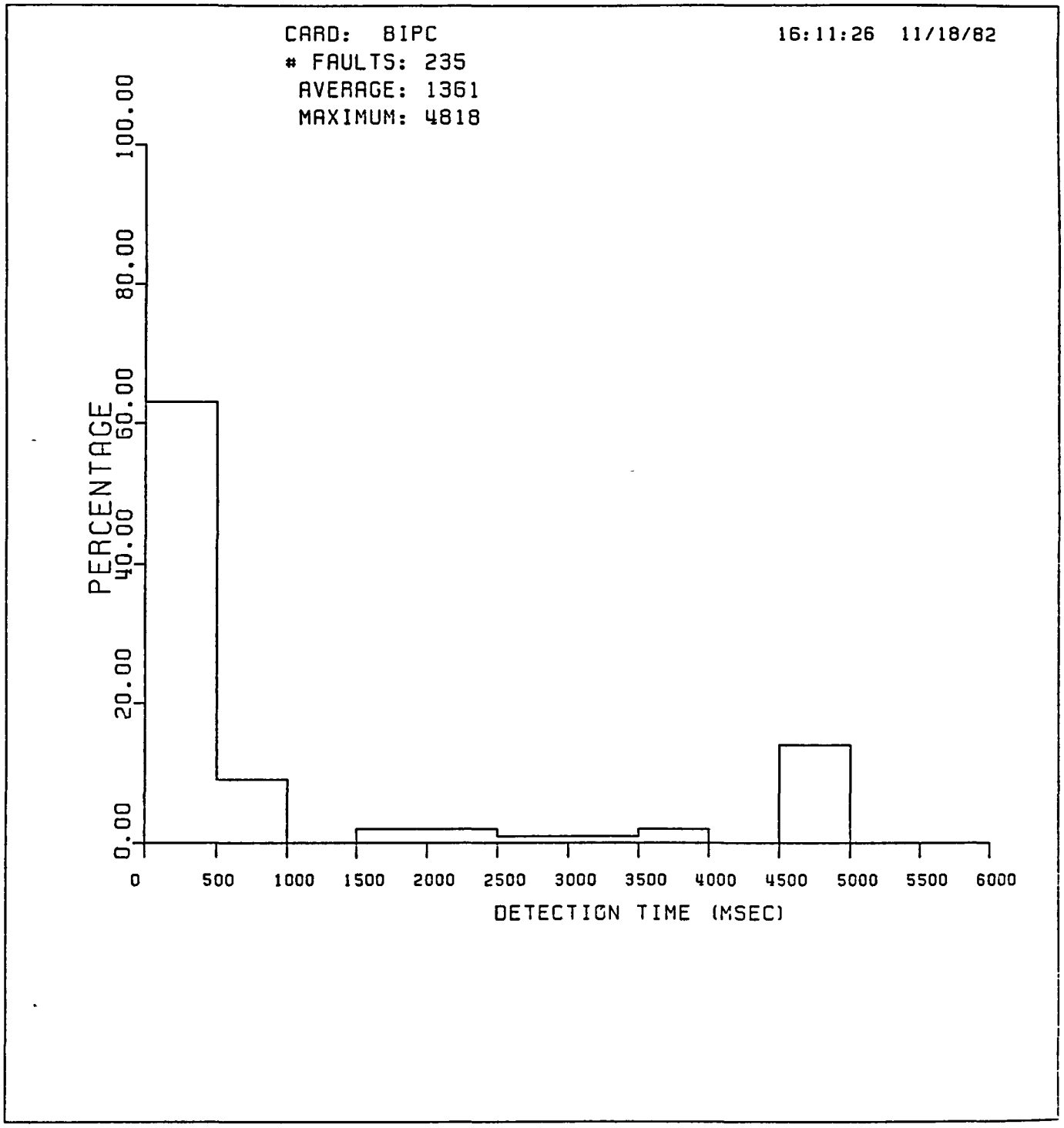
CARD: BIPC
16:11:26 11/18/82
FAULTS: 235
AVERAGE: 1361
MAXIMUM: 4818
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
DETECTION TIME (MSEC)

Figure 42

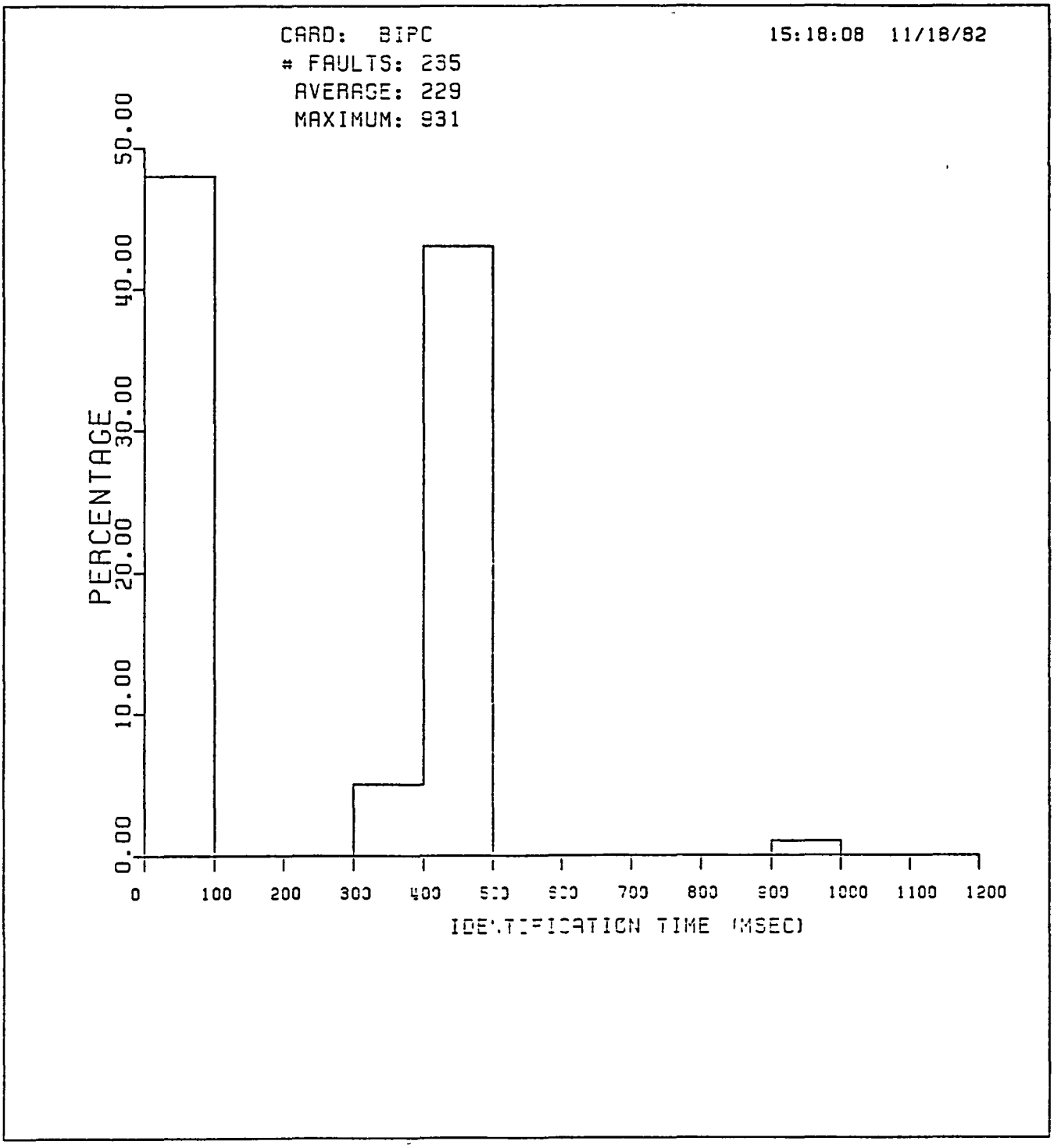
CARD: BIPC
15:18:08 11/18/82
FAULTS: 235
AVERAGE: 229
MAXIMUM: 931
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
IDENTIFICATION TIME (MSEC)

Figure 43

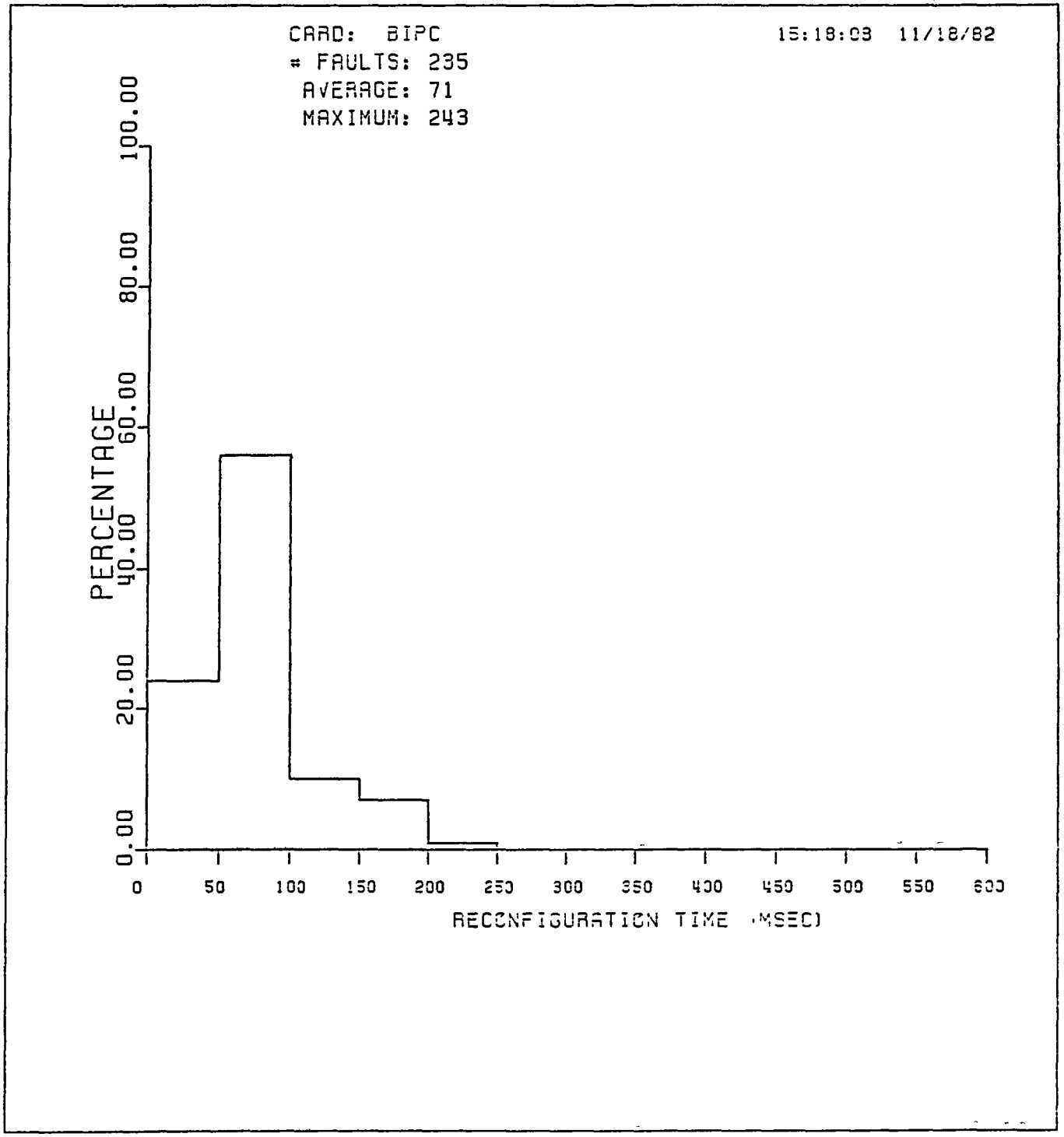
CARD: BIPC
15:18:03 11/18/82
FAULTS: 235
AVERAGE: 71
MAXIMUM: 243
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 44

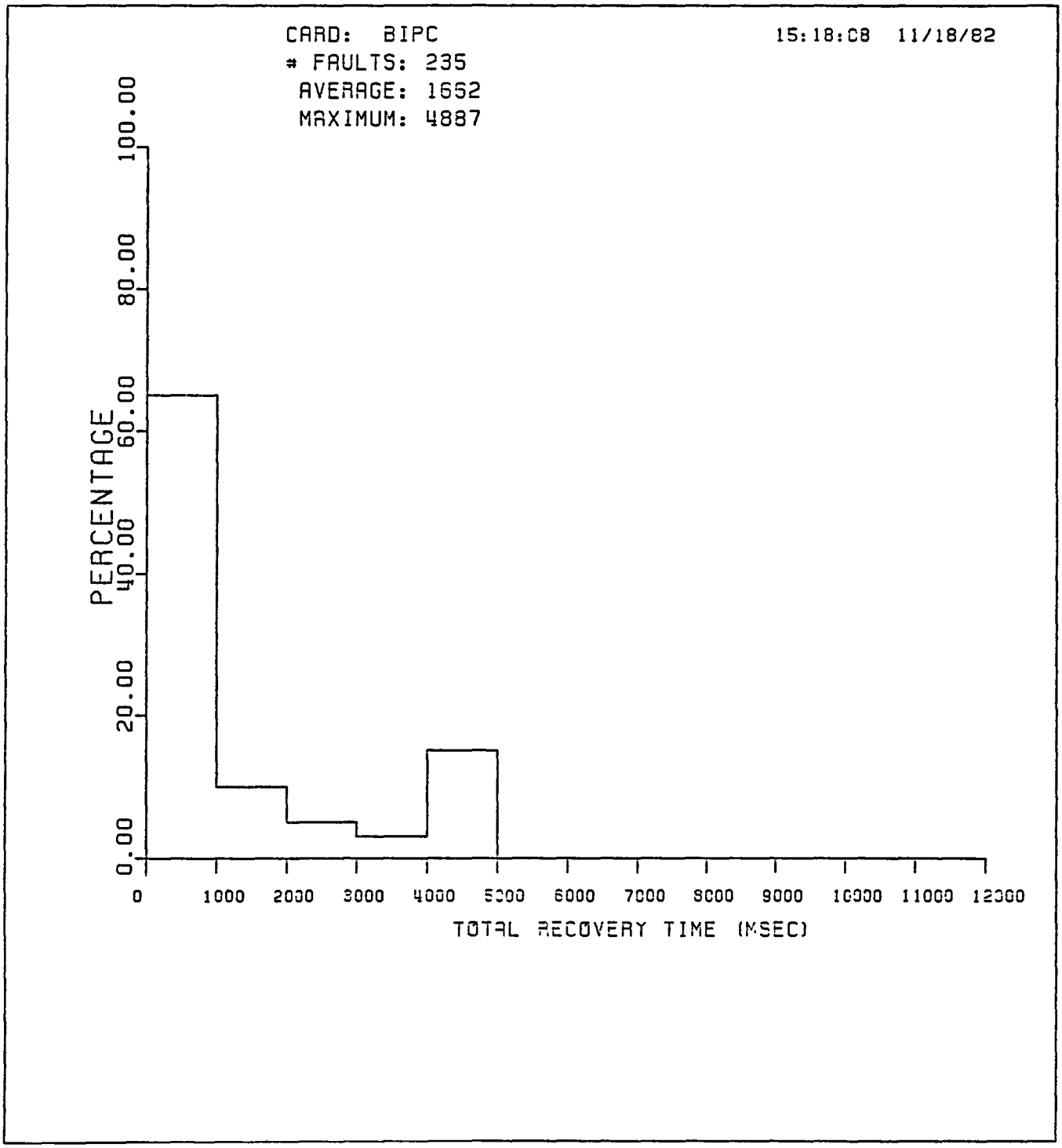
CARD: BIPC
15:18:08 11/18/82
FAULTS: 235
AVERAGE: 1652
MAXIMUM: 4887
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
TOTAL RECOVERY TIME (MSEC)

Figure 45

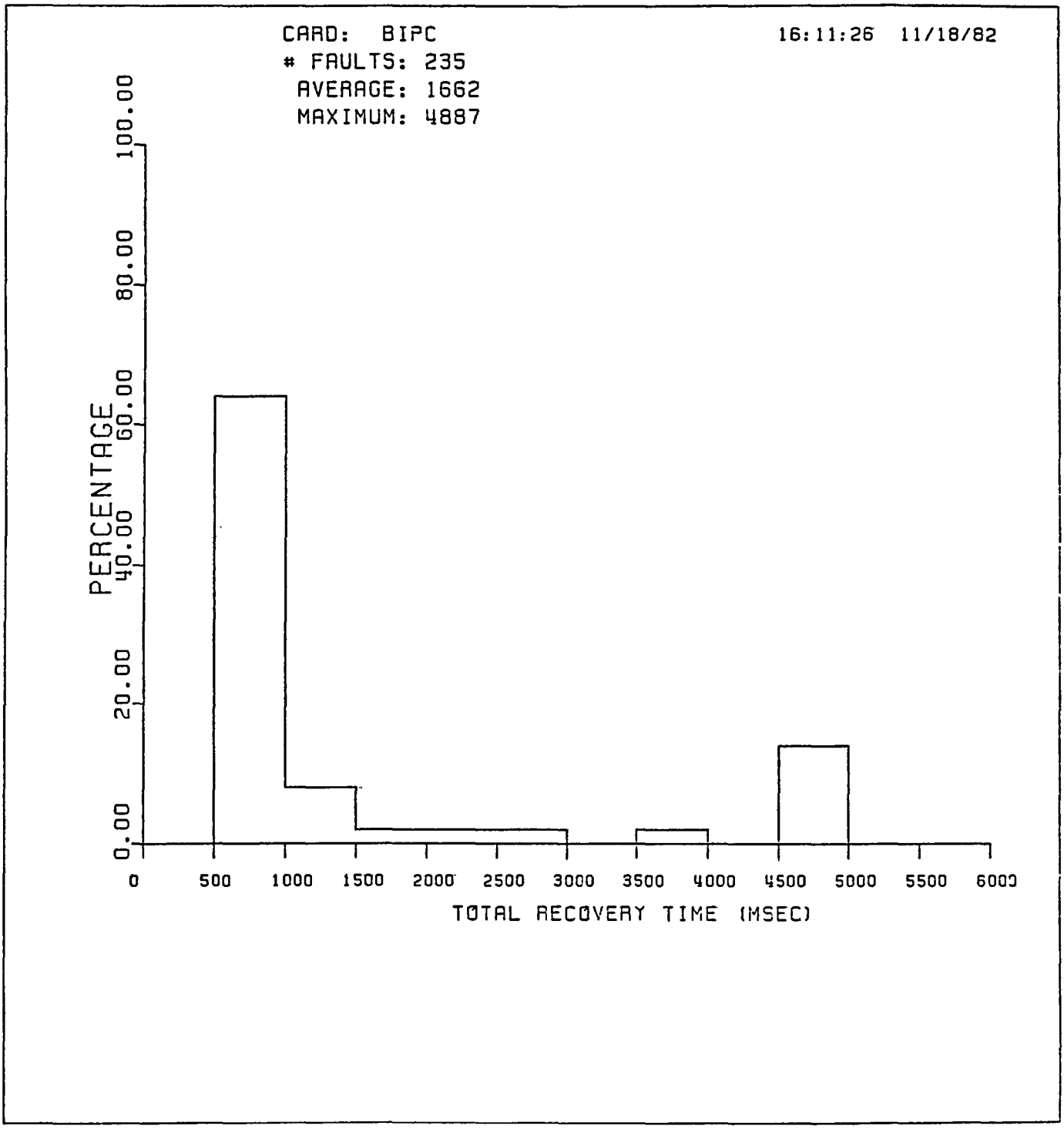
CARD: BIPC
16:11:26 11/18/82
FAULTS: 235
AVERAGE: 1662
MAXIMUM: 4887
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
TOTAL RECOVERY TIME (MSEC)

Figure 46

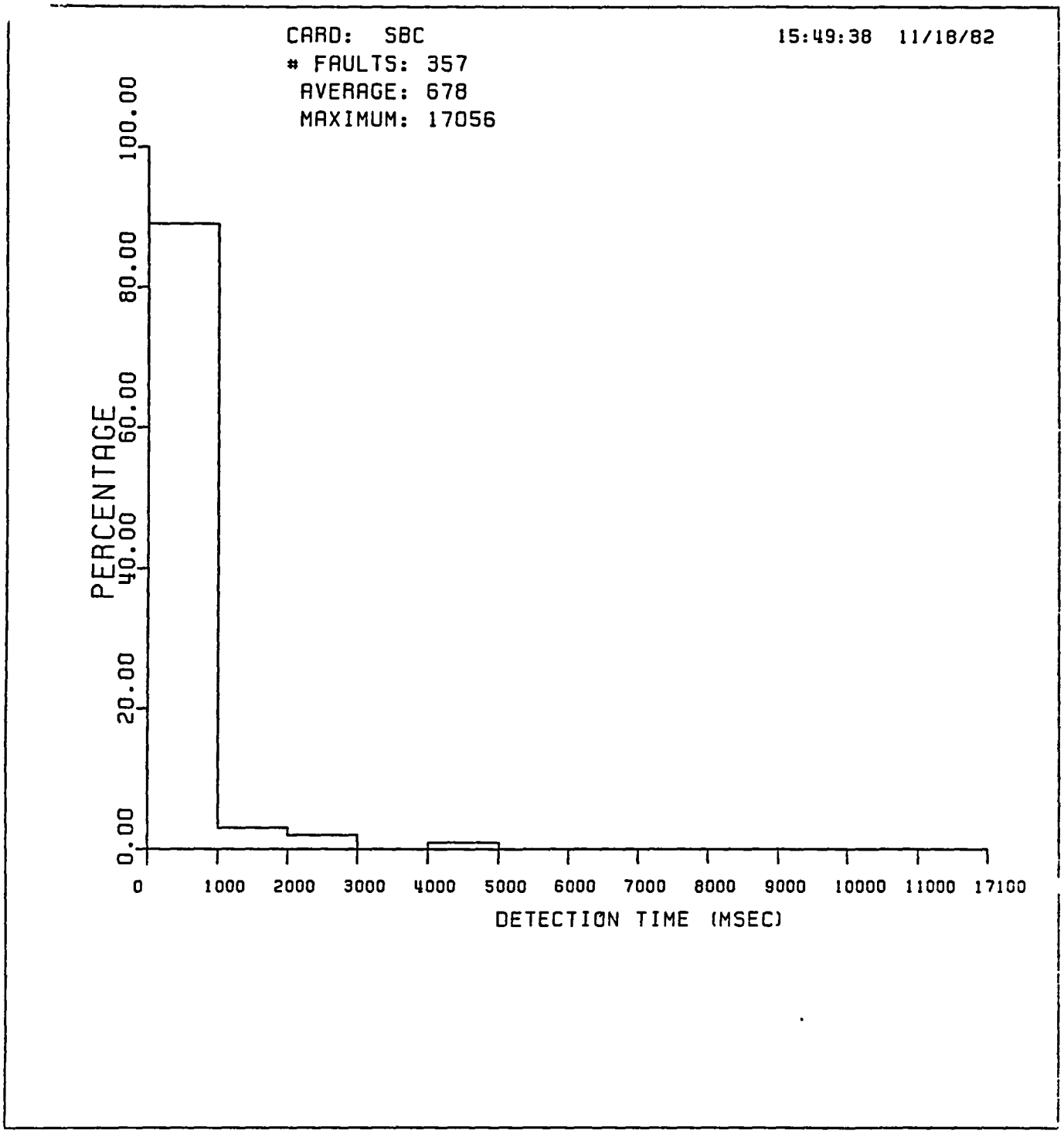
CARD: SBC
15:49:38 11/18/82
FAULTS: 357
AVERAGE: 678
MAXIMUM: 17056
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
17100
DETECTION TIME (MSEC)

Figure 47

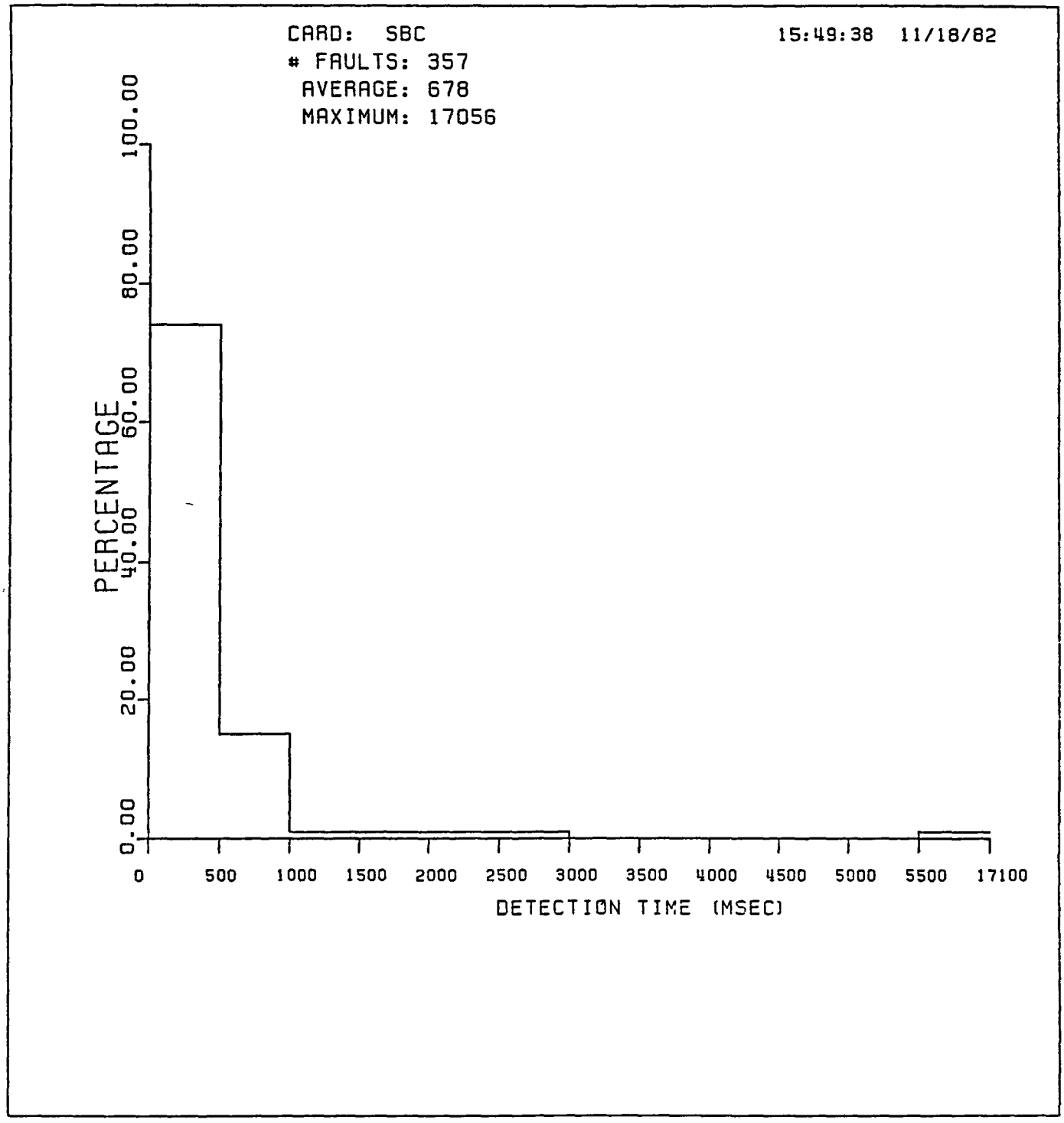
CARD: SBC
15:49:38 11/18/82
FAULTS: 357
AVERAGE: 678
MAXIMUM: 17056
PERCENTAGE
0.00
20.00
40.00
60.00
80.00
100.00
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
17100
DETECTION TIME (MSEC)

Figure 48

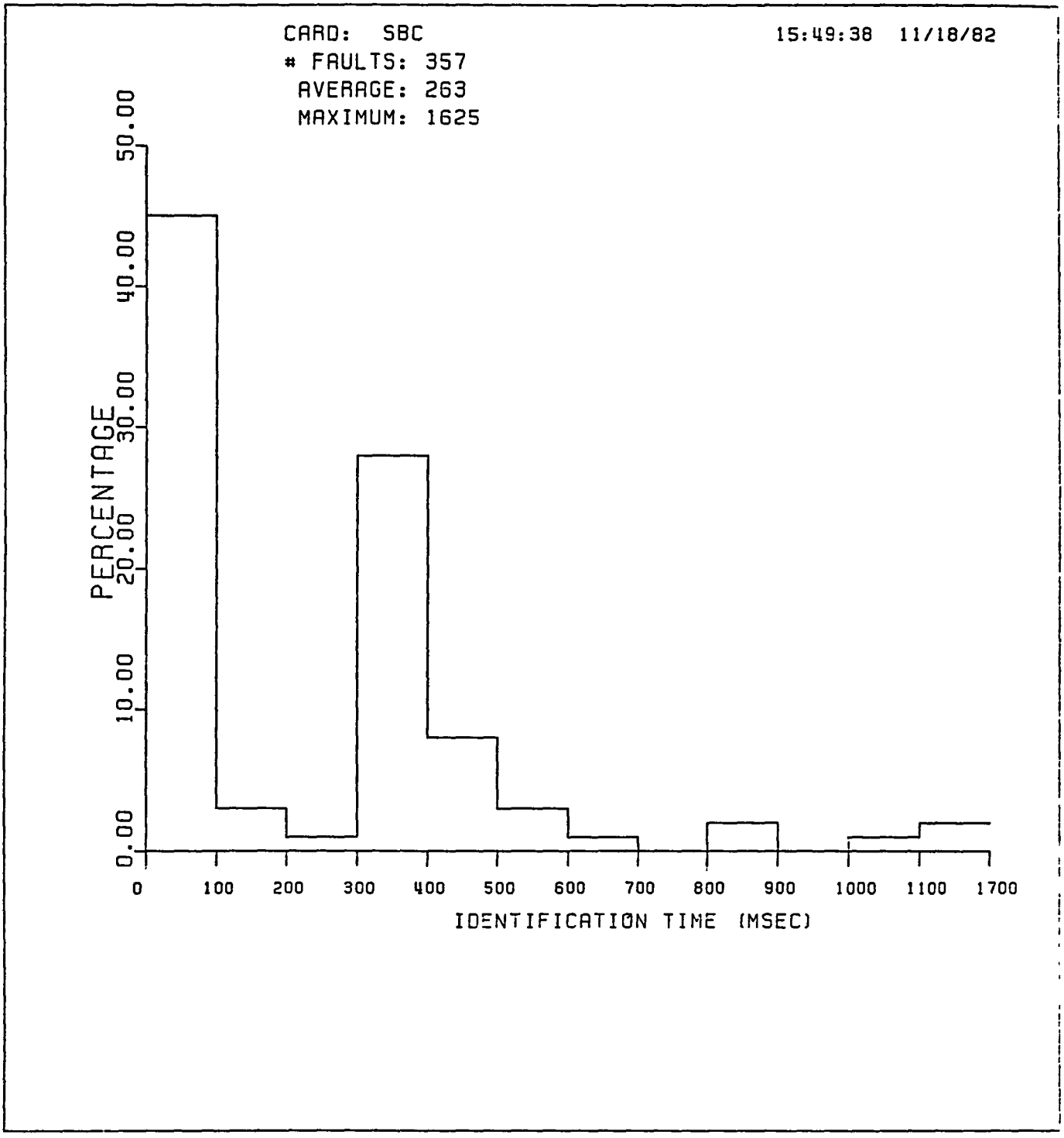
CARD: SBC
15:49:38 11/18/82
FAULTS: 357
AVERAGE: 263
MAXIMUM: 1625
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1700
IDENTIFICATION TIME (MSEC)

Figure 49

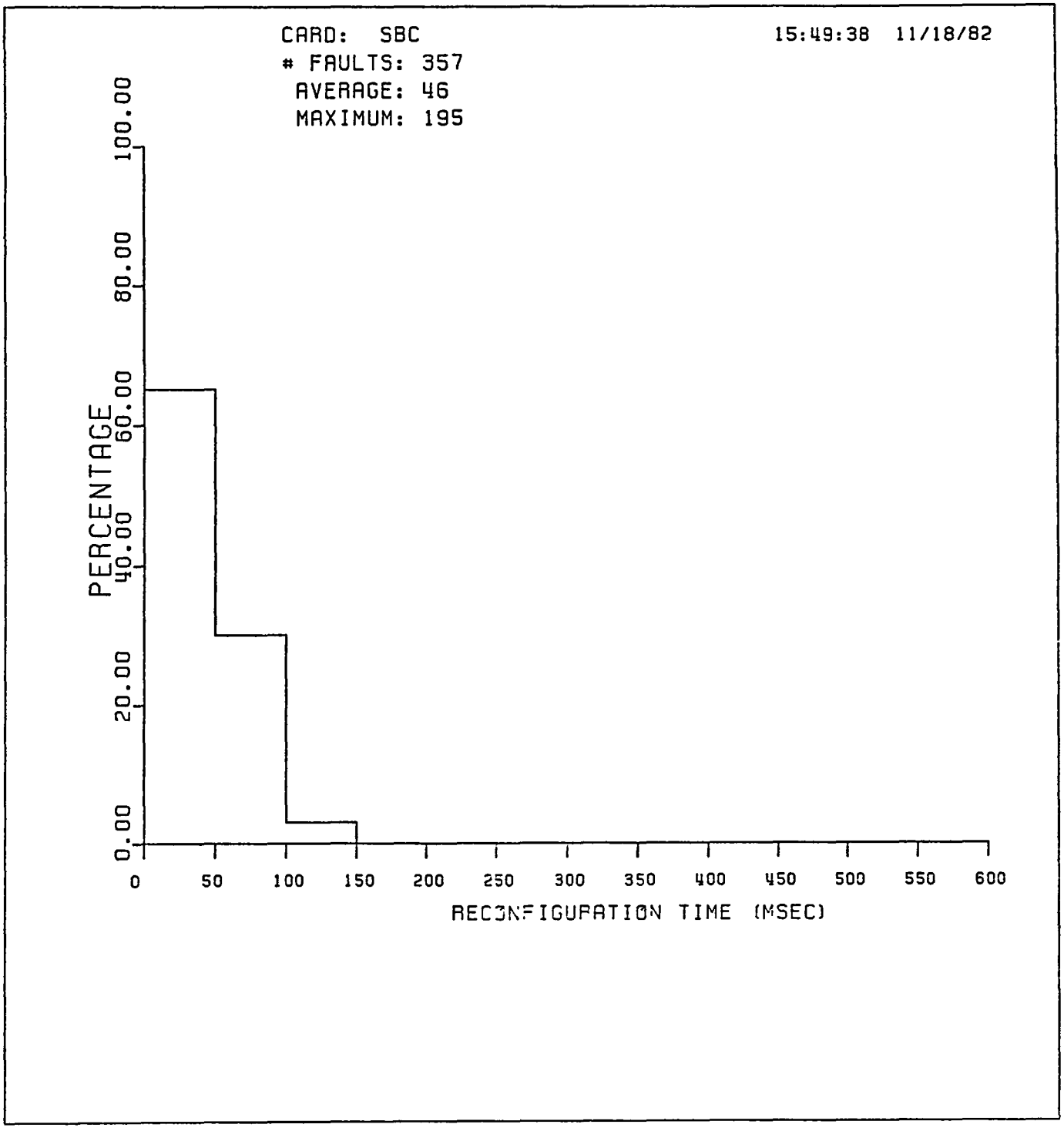
CARD: SBC
15:49:38 11/18/82
FAULTS: 357
AVERAGE: 46
MAXIMUM: 195
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 50

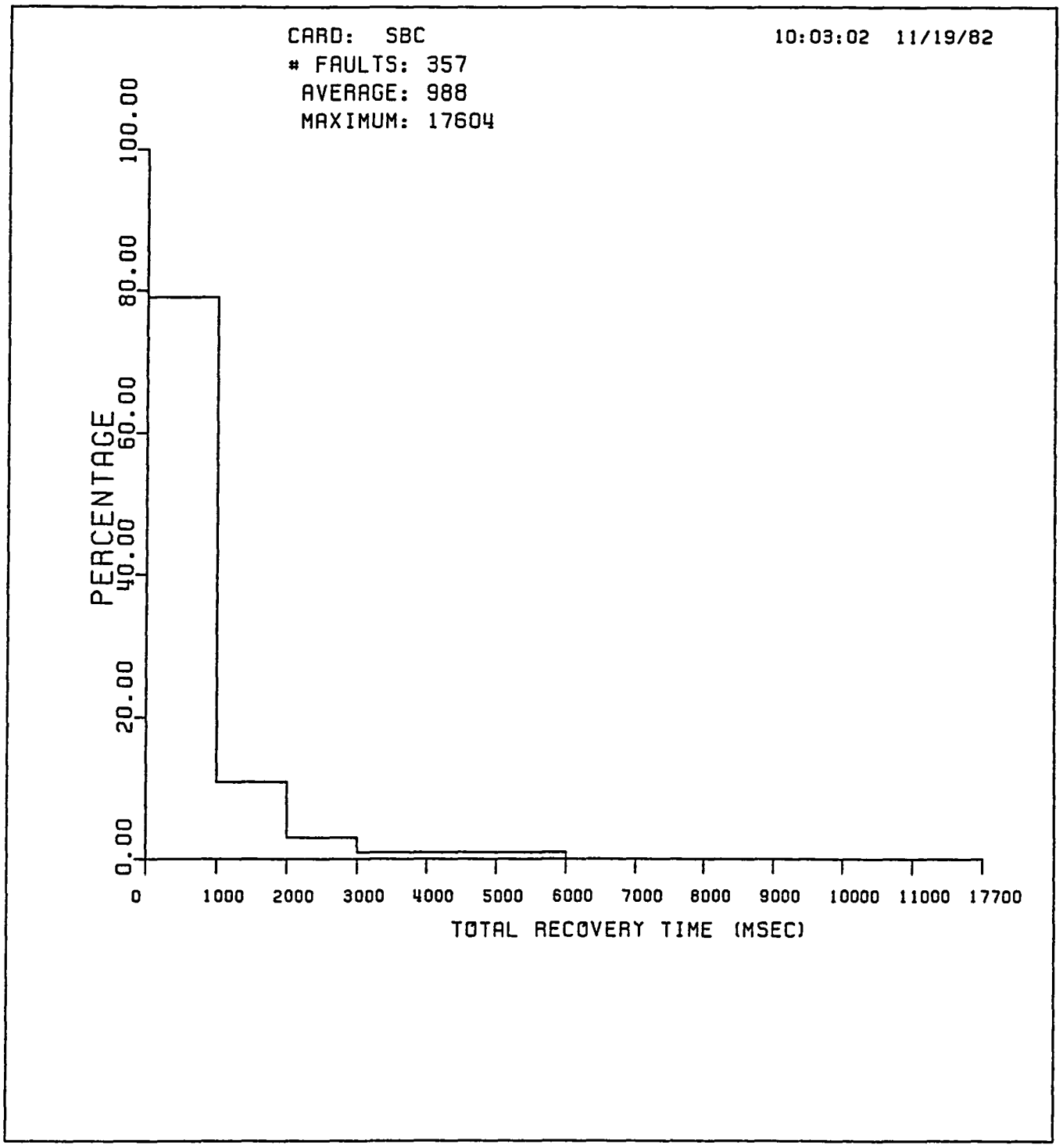
CARD: SBC
10:03:02 11/19/82
FAULTS: 357
AVERAGE: 988
MAXIMUM: 17604
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
17700
TOTAL RECOVERY TIME (MSEC)

Figure 51

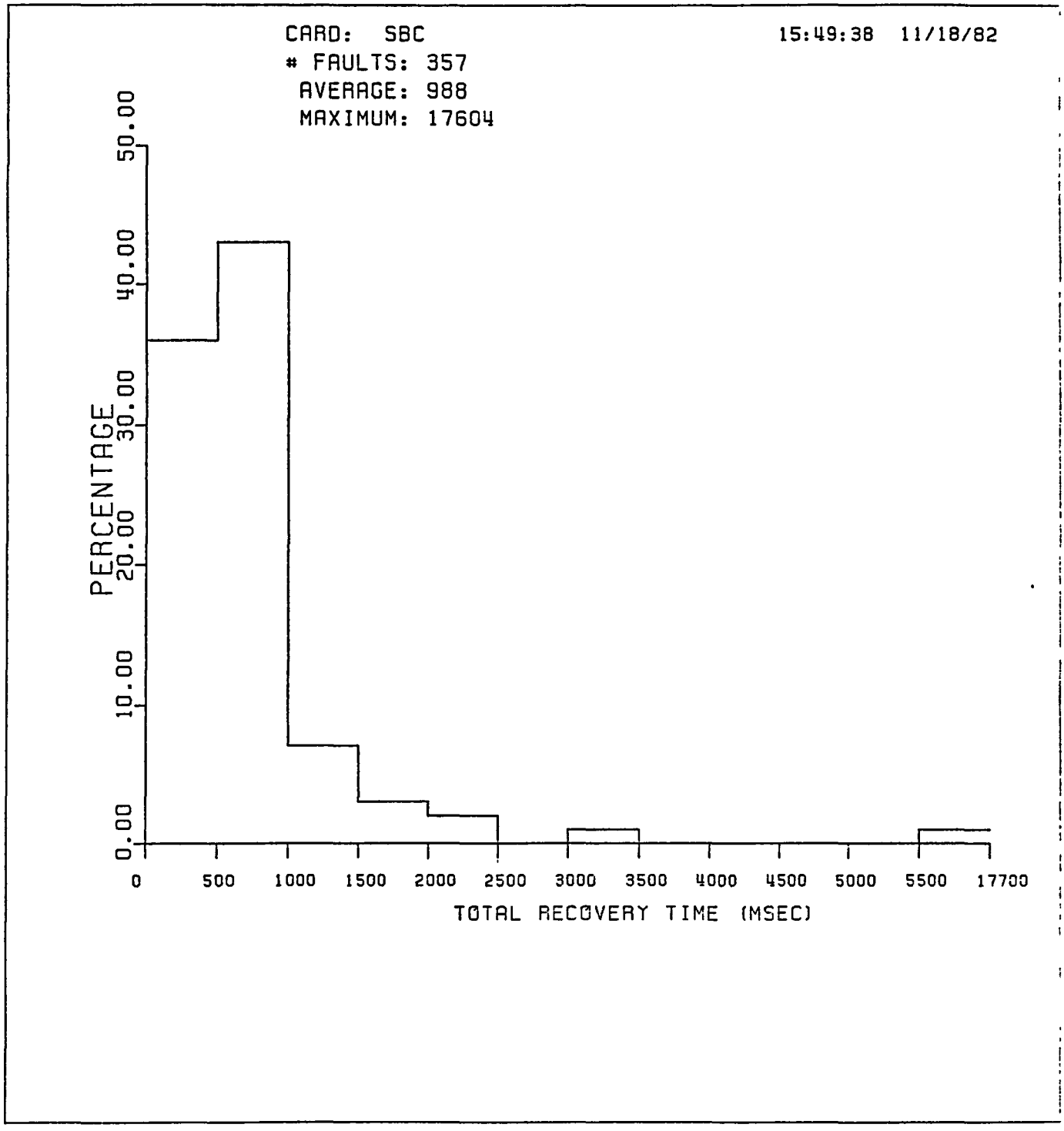
CARD: SBC
15:49:38 11/18/82
FAULTS: 357
AVERAGE: 988
MAXIMUM: 17604
PERCENTAGE
0.00
10.00
20.00
30.00
40.00
50.00
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
17700
TOTAL RECOVERY TIME (MSEC)

Figure 52

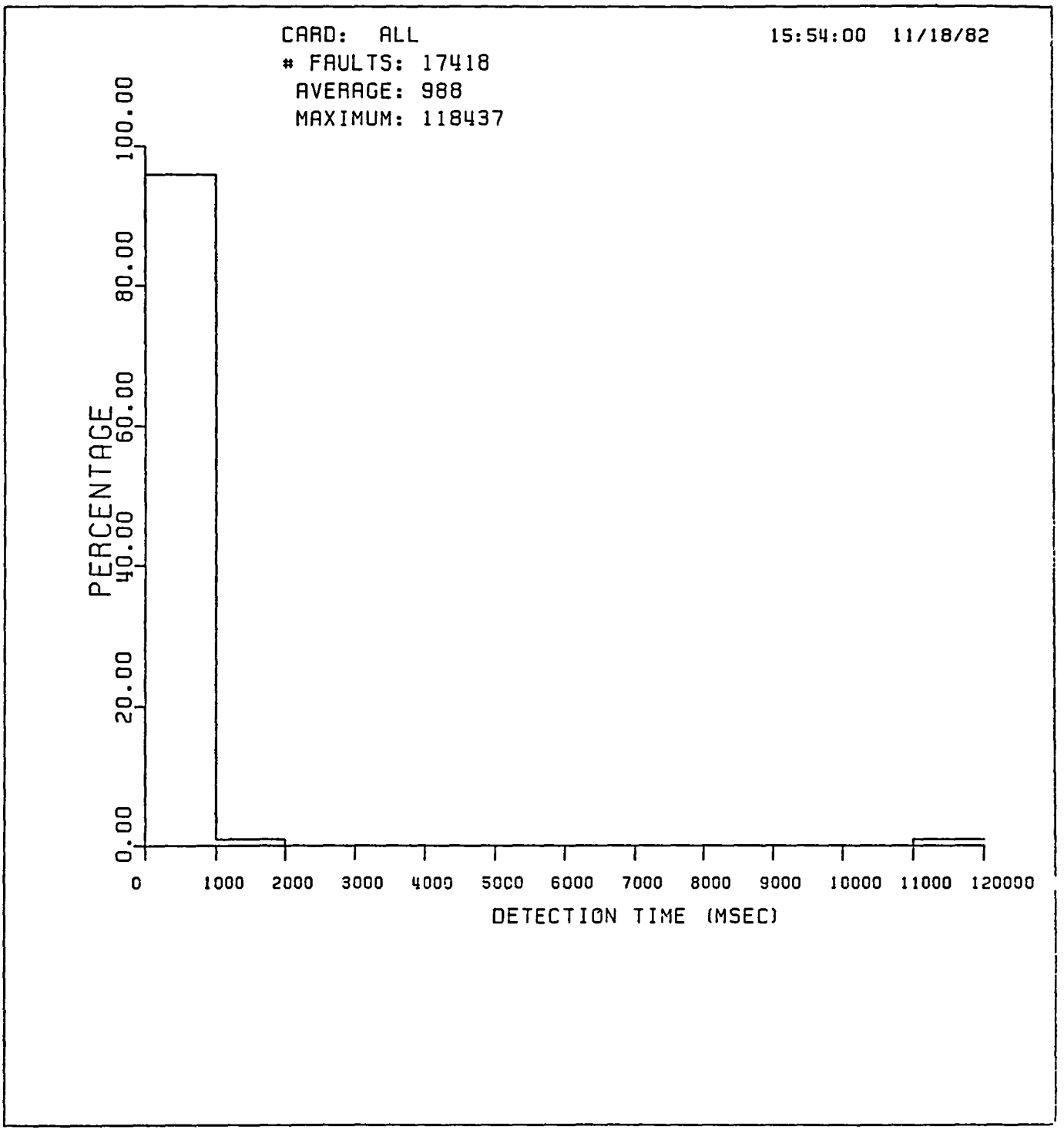
CARD: ALL
15:54:00 11/18/82
FAULTS: 17418
AVERAGE: 988
MAXIMUM: 118437
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
120000
DETECTION TIME (MSEC)

Figure 53

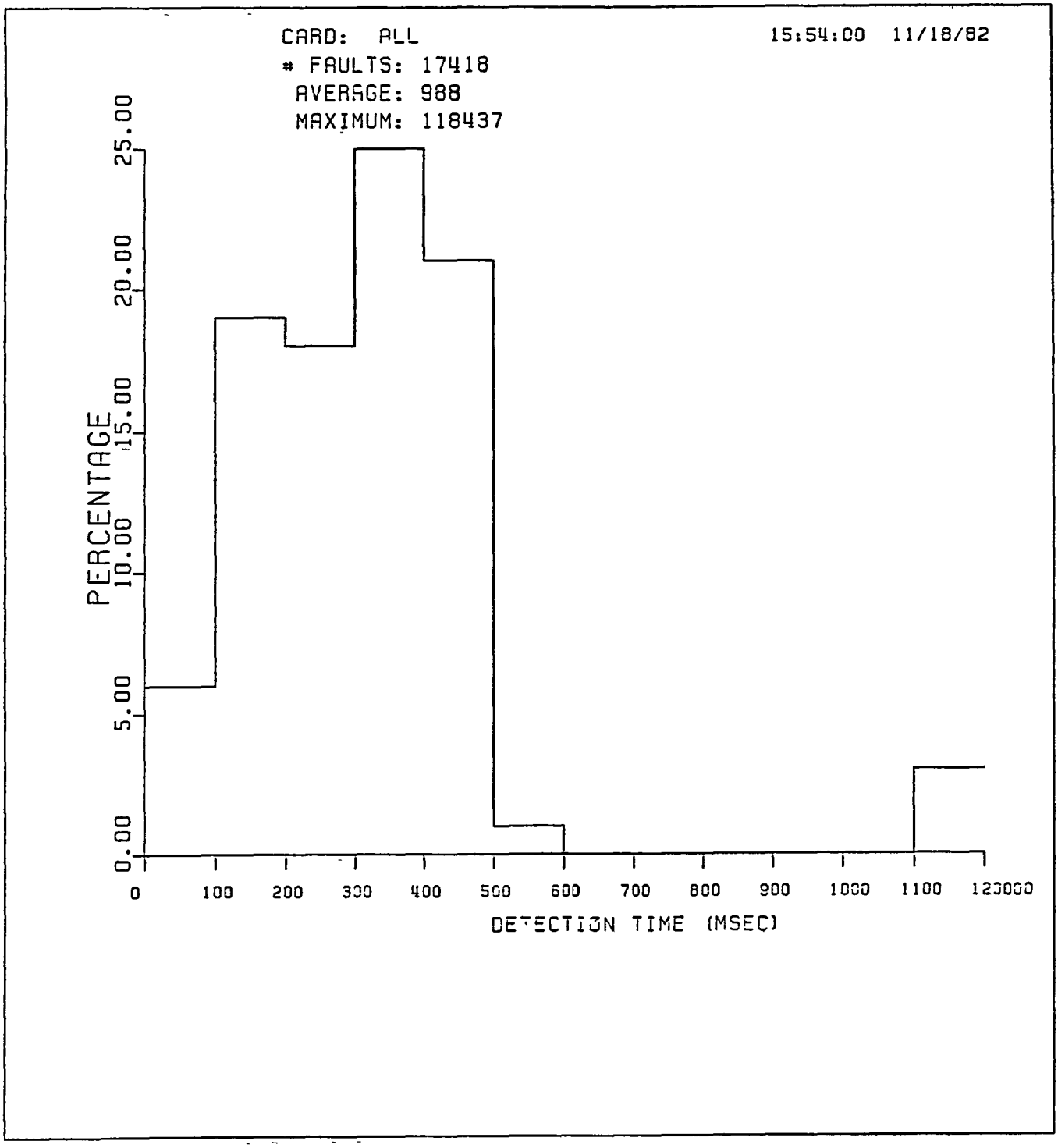
CARD: ALL
15:54:00 11/18/82
FAULTS: 17418
AVERAGE: 988
MAXIMUM: 118437
PERCENTAGE
0.00
5.00
10.00
15.00
20.00
25.00
0
100
200
300
400
500
600
700
800
900
1000
1100
120000
DETECTION TIME (MSEC)

Figure 54

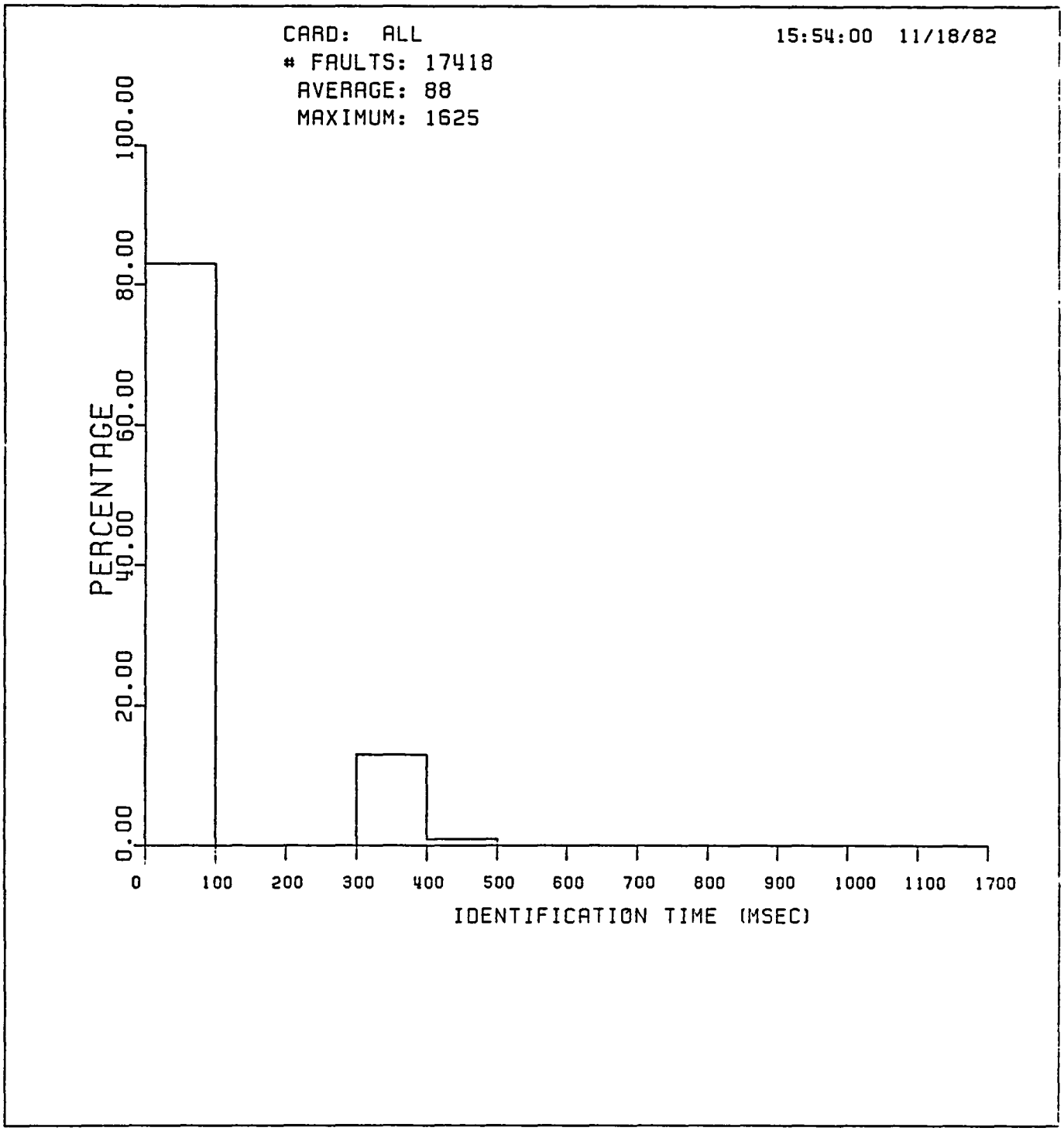
CARD: ALL
15:54:00 11/18/82
FAULTS: 17418
AVERAGE: 88
MAXIMUM: 1625
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
1700
IDENTIFICATION TIME (MSEC)

Figure 55

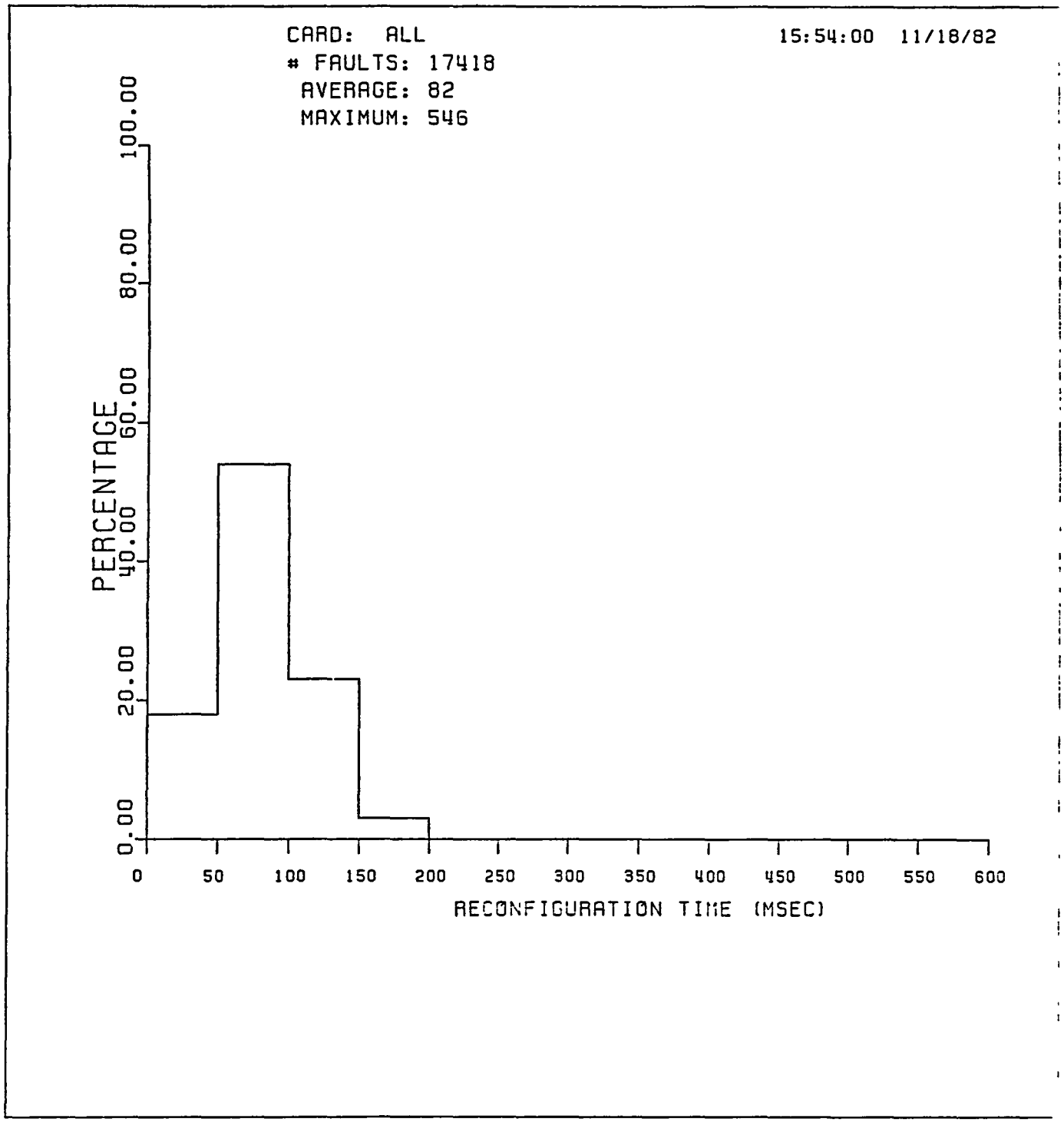
CARD: ALL
15:54:00 11/18/82
FAULTS: 17418
AVERAGE: 82
MAXIMUM: 546
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
50
100
150
200
250
300
350
400
450
500
550
600
RECONFIGURATION TIME (MSEC)

Figure 56

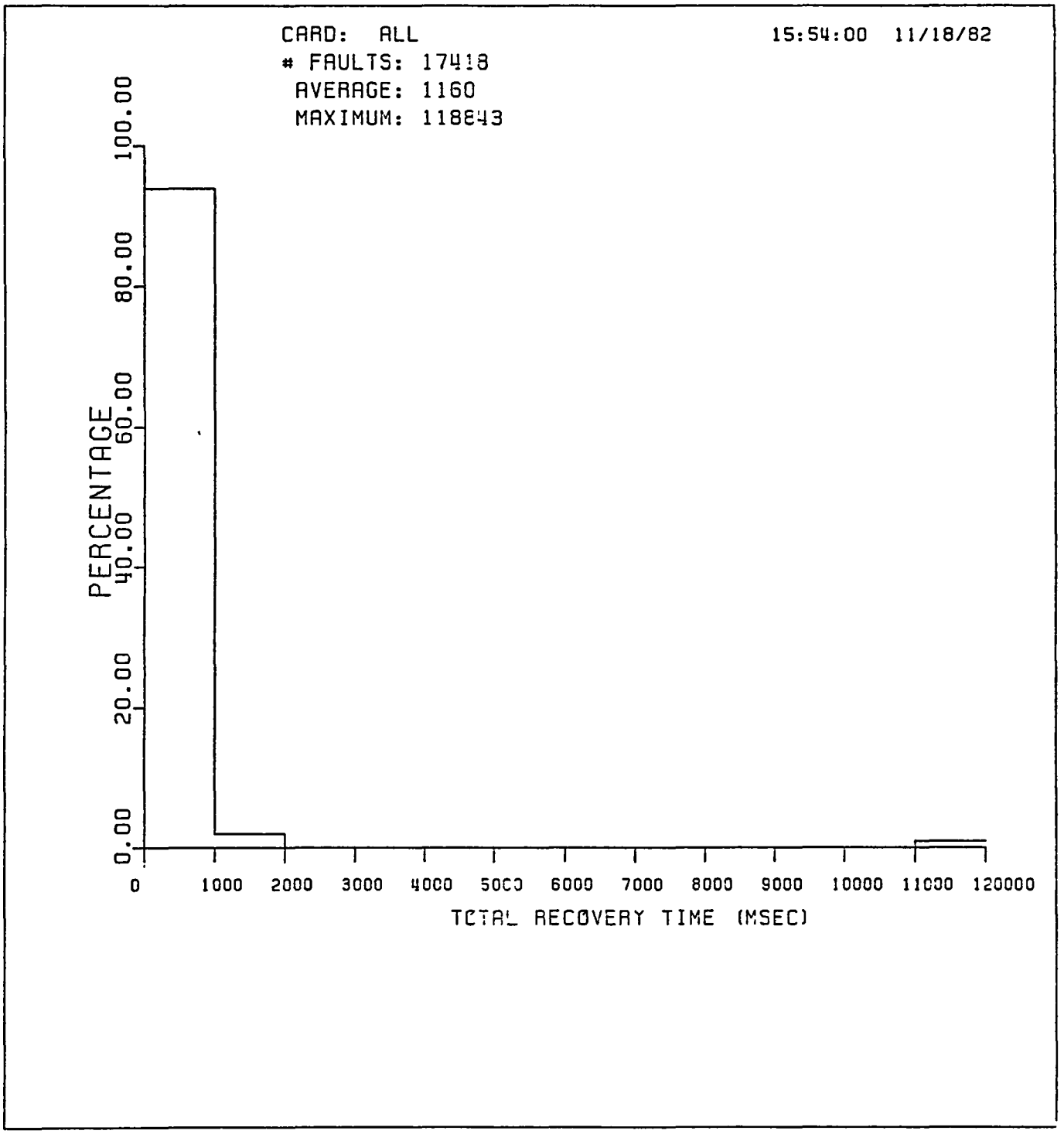
CARD: ALL
15:54:00 11/18/82
FAULTS: 17418
AVERAGE: 1160
MAXIMUM: 118843
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
120000
TOTAL RECOVERY TIME (MSEC)

Figure 57

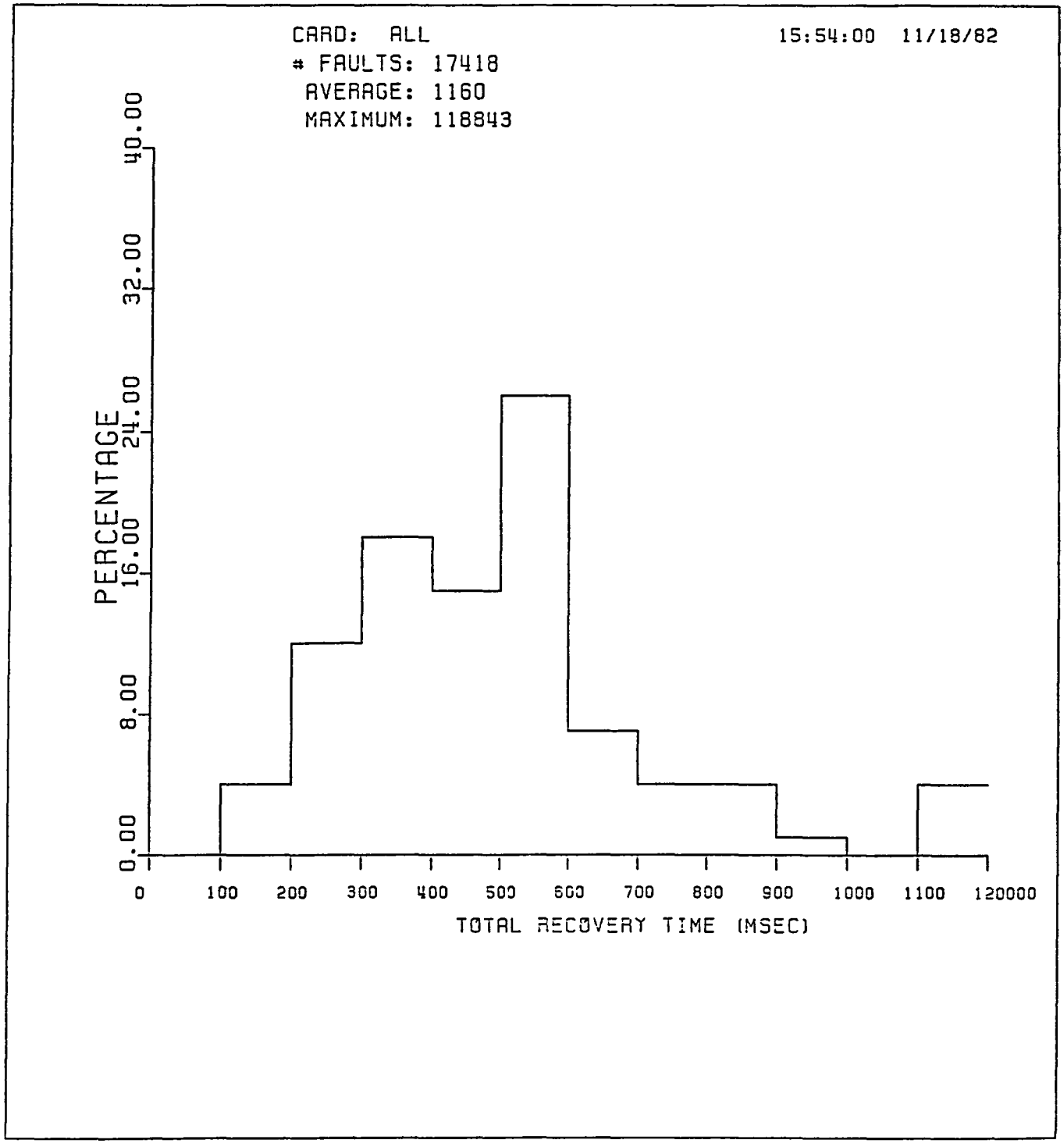
CARD: ALL
15:54:00 11/18/82
FAULTS: 17418
AVERAGE: 1160
MAXIMUM: 118843
PERCENTAGE
0.00
8.00
16.00
24.00
32.00
40.00
0
100
200
300
400
500
600
700
800
900
1000
1100
120000
TOTAL RECOVERY TIME (MSEC)

Figure 58

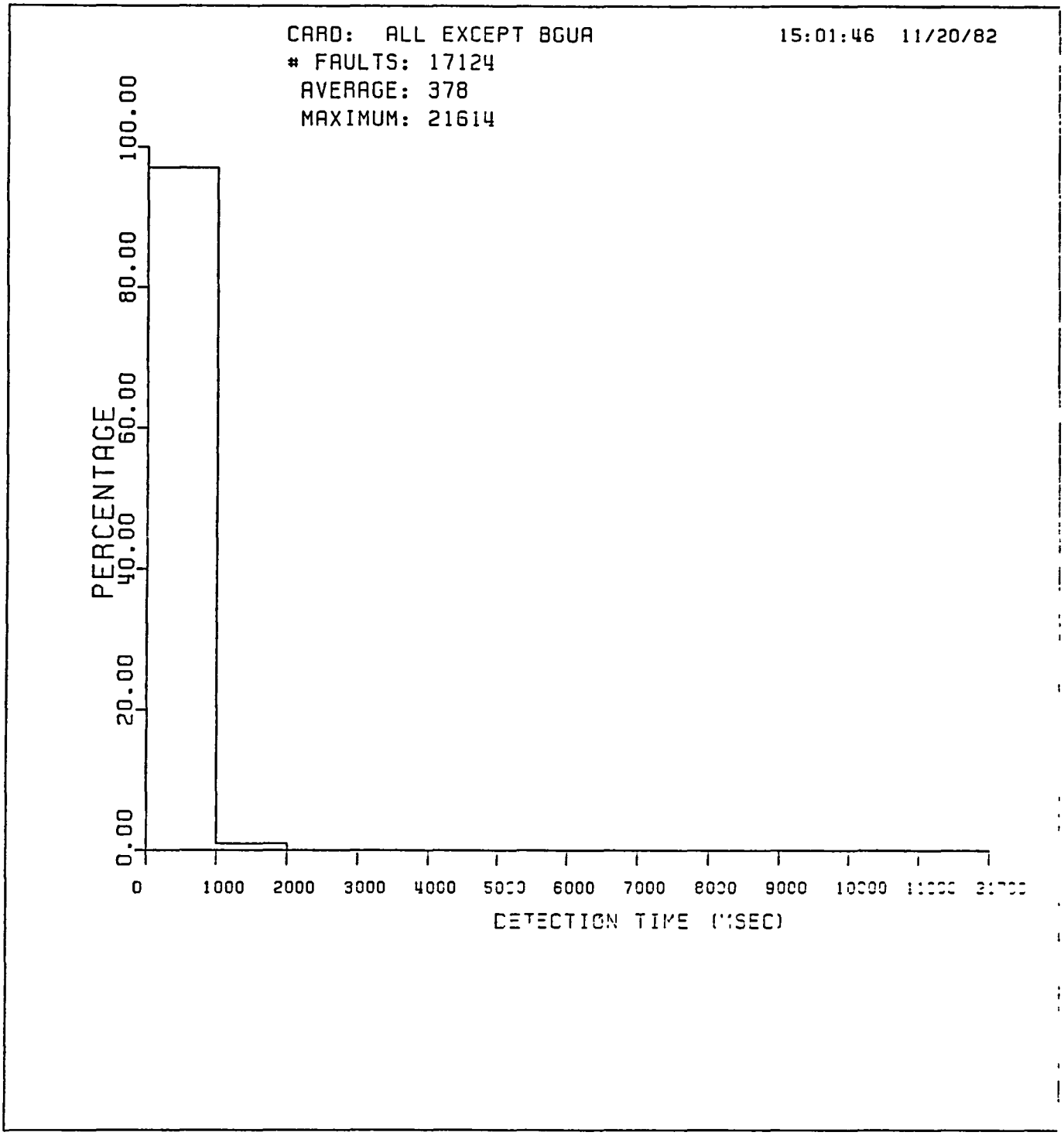
CARD: ALL EXCEPT BGUA
15:01:46 11/20/82
FAULTS: 17124
AVERAGE: 378
MAXIMUM: 21614
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
DETECTION TIME (MSEC)

Figure 59

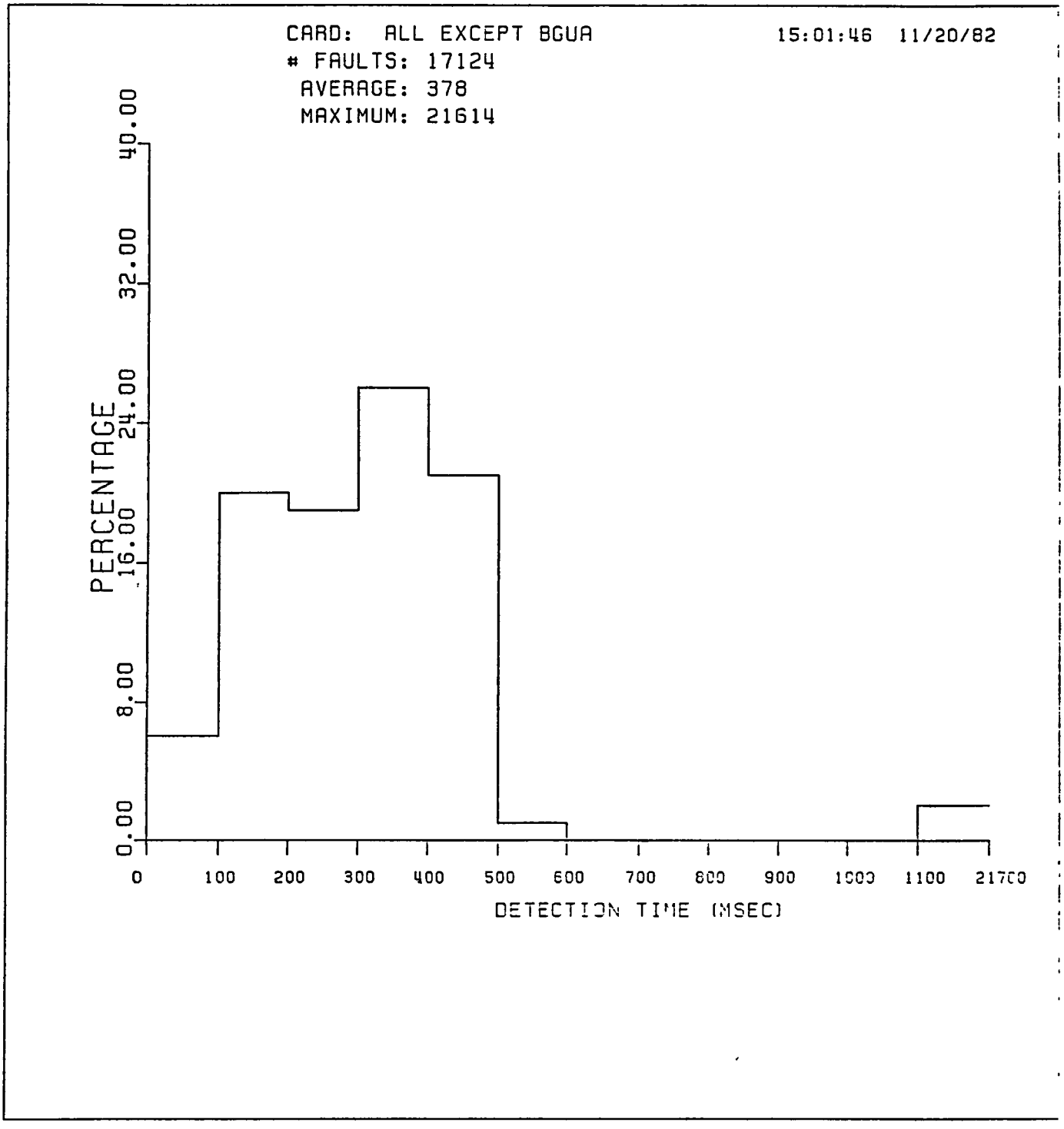
CARD: ALL EXCEPT BGUA
15:01:46 11/20/82
FAULTS: 17124
AVERAGE: 378
MAXIMUM: 21614
PERCENTAGE
40.00
32.00
24.00
16.00
8.00
0.00
0
100
200
300
400
500
600
700
800
900
1000
1100
21700
DETECTION TIME (MSEC)

Figure 60

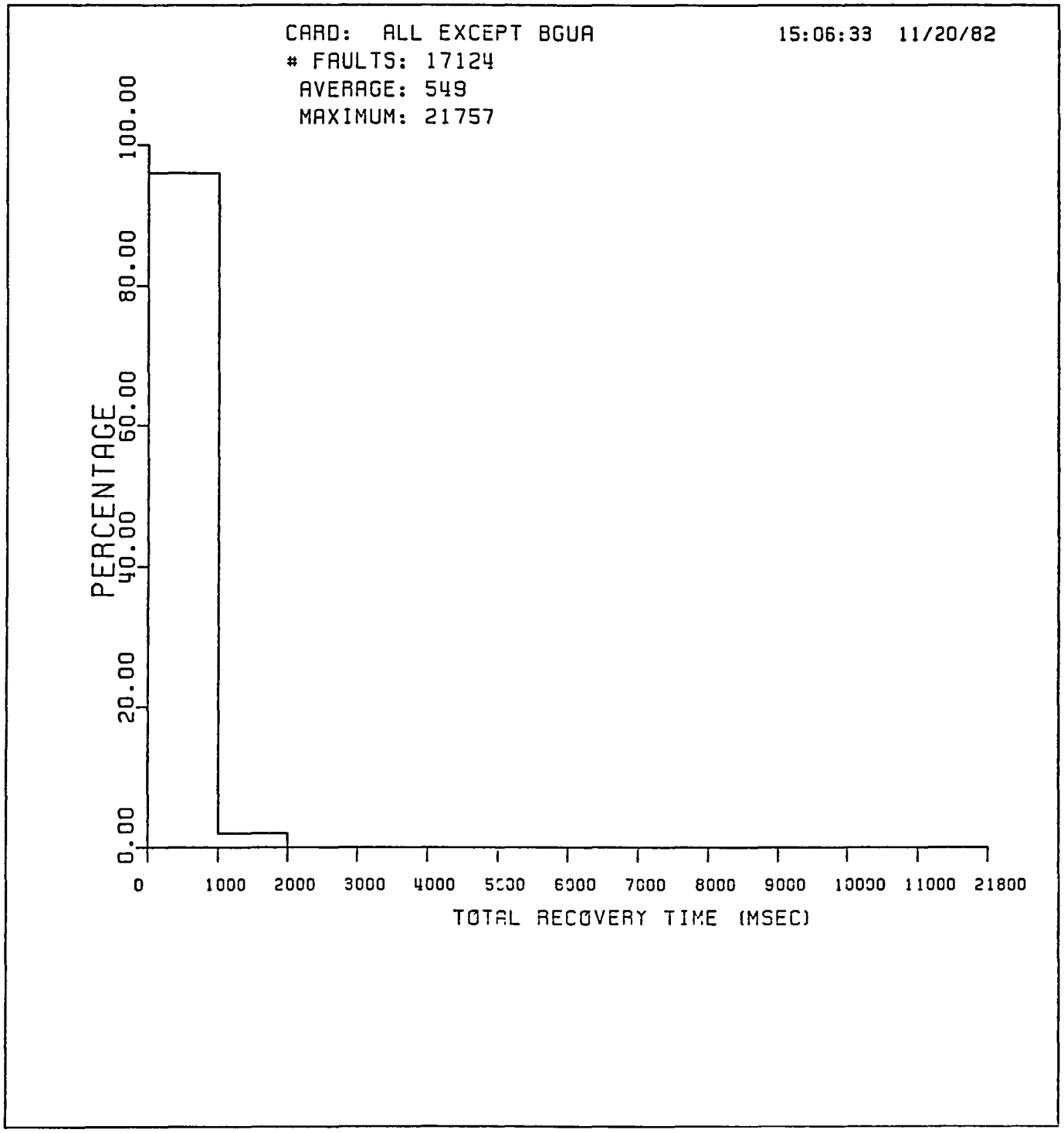
CARD: ALL EXCEPT BGUA
15:06:33 11/20/82
FAULTS: 17124
AVERAGE: 549
MAXIMUM: 21757
PERCENTAGE
100.00
80.00
60.00
40.00
20.00
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
21800
TOTAL RECOVERY TIME (MSEC)

Figure 61

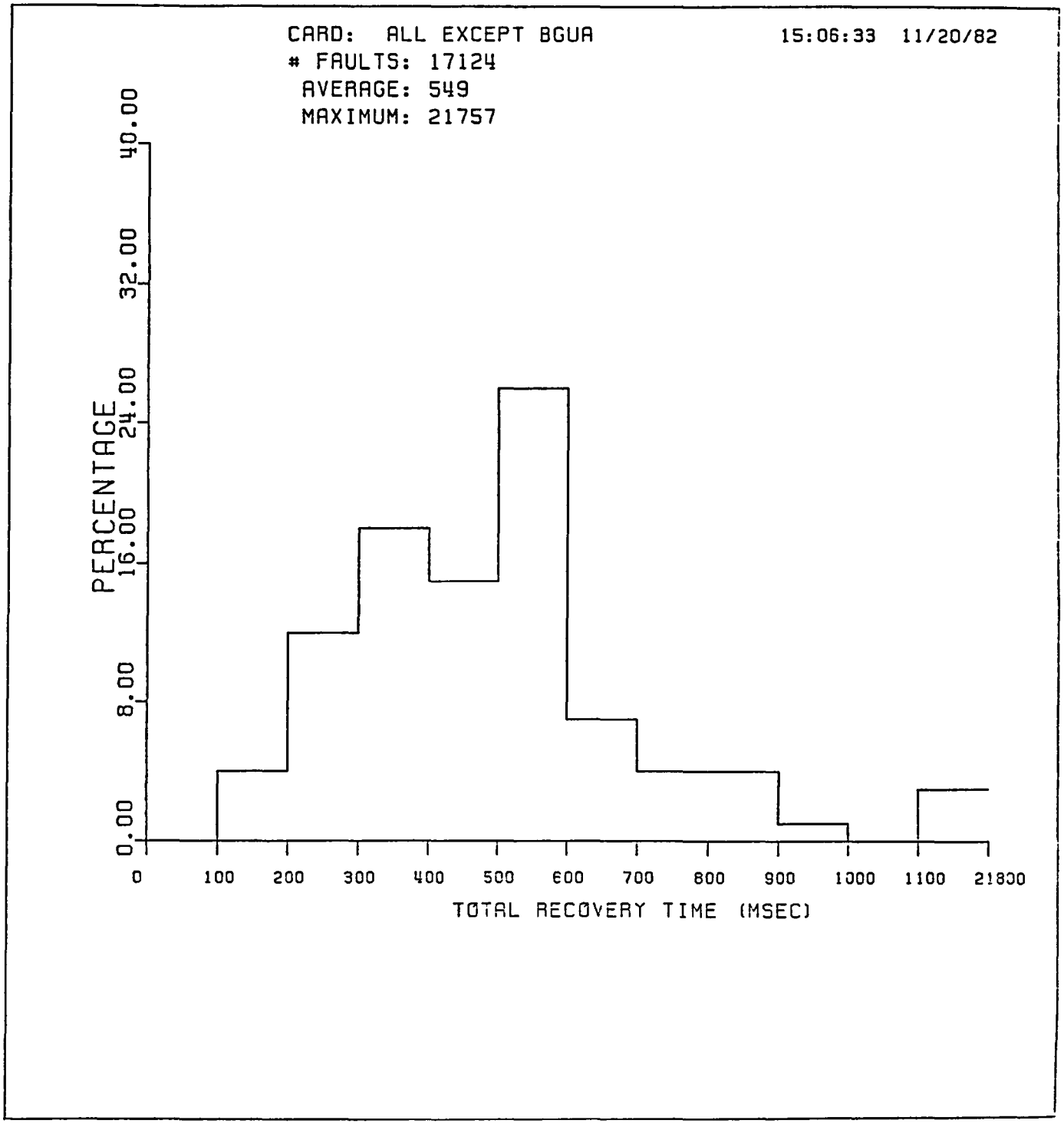
CARD: ALL EXCEPT BGUA
15:06:33 11/20/82
FAULTS: 17124
AVERAGE: 549
MAXIMUM: 21757
PERCENTAGE
0.00
8.00
16.00
24.00
32.00
40.00
0
100
200
300
400
500
600
700
800
900
1000
1100
21800
TOTAL RECOVERY TIME (MSEC)

Figure 62

two times for most cards. For instance, Figure 8 shows this variable for CPUD using a bucket size of 1 second. This scale allows the maximum detection time (9.1 seconds for this card) to be accommodated. But since 98 percent of all faults are uncovered within a second, a lot of information is lost. Therefore this same data is replotted in Figure 9 using a bucket size of 100 milliseconds. All detection times longer than a second are lumped together at the end of the plot. This figure shows in much greater detail the distribution of 98 percent of detection times. Finally, the total recovery time is also plotted several times for each card using different scales. For example, Figures 12 and 13 show the probability density function of this variable for CPUD with scales of 1 second and 0.1 second, respectively. The contents of Figures 8-62 will be discussed next.

It may be observed that there is a great variation in pdf's of detection, identification, and reconfiguration times. But there is not much variation amongst cards for any given parameter. For instance, identification time probability density function for CPUD (Figure 10) is very similar to that for PROM (Figure 21) or BGU (Figure 34). But it is quite different from the detection time pdf for the same card (Figures 8, 9). Characteristics of pdf's for each of the four parameters in general rather than for each card will therefore be discussed next. Exceptions where appropriate will be pointed out.

Probability density functions of detection time reveal the complexity of the detection phase. Over 95 percent of the faults for

almost all the cards are detected within 600 milliseconds or two R1 frames. For CPU data (Figures 8, 9) and control cards (Figures 14, 15), this figure rises to almost 99 percent. The latency in reading error latches varies from 0 to 340 milliseconds (one R1 frame) depending on when the fault was injected with respect to the beginning of the SCC task. Evidently, not all faults manifest themselves as bus errors right away. For instance, about 20 percent of the faults injected in the CPU data card are uncovered between 400 and 600 milliseconds. Evidently it took these faults between 100 to 300 msecs to cause erroneous data to appear on the buses. But these faults are uncovered by routine programs.

Beyond this initial impulse, there is a long tail in the detection time pdf that goes out to about 20 seconds. This corresponds to faults that are only uncovered by self-test programs. The fraction of faults that falls under this long tail is only about 2 to 4 percent for the processor region cards (CPUD - Figure 8, CPUC - Figure 14, PROM - Figure 20, Cache Controller - Figure 27). For the bus interface cards this figure is much higher, as seen in pdf plots for BIT (Figure 37), BIPC (Figure 41), and SBC (Figure 48). This is due to the fact that faults on these boards were deliberately concentrated into error detection and masking circuitry. Most of these faults require self-test programs to be uncovered. There is a lot of other random logic on these boards that was not subjected to faults, and it is most likely that faults into these circuits would be detected by routine program execution. As can be seen from the number of faults injected into each board, the processor region cards have been much more thoroughly tested than the bus interface and

control cards. (Only about 1,000 out of 21,000 were injected into the latter.) If they were tested as completely as the processor region, their detection time distribution would tend to be closer to that for the processor region.

The detection time pdf for BGU (Figures 32, 33) is totally different from all others with a very high number of faults being detected between 40 and 50 seconds and the maximum going out to 2 minutes. This, as explained earlier, is due to the fact that BGU faults were detected by normal system reconfiguation, a complete cycle of which takes 6 minutes. Self-test programs for this part of the FTMP would decrease the BGU detection times by an order of magnitude.

Detection time pdf for all 17,418 faults is shown in Figures 53 and 54. About 96 percent of all faults are detected in 600 milliseconds or less. The detection time distribution for all faults except the BGU faults shown in Figures 59 and 60 looks very much the same. The only difference appears in the average detection time, which drops from 988 to only 378 milliseconds. This latter figure is more representative of what may be expected as the FTMP response if a reasonable set of diagnostic programs had been completed.

The next parameter is the identification time. As discussed earlier in this chapter, fault identification is a deterministic phase of the recovery procedure. For a given pin fault and a given system configuration one can say with certainty as to how many passes of identification program are required to isolate the faulty unit. This is borne out by the probability density functions for identification time. As seen

in Figure 10, about 85 percent of CPUD faults are identified between 0 and 100 milliseconds. Most of the remaining faults are identified between 300 and 400 milliseconds. It will be recalled here that the identification program runs every 320 milliseconds. What this pdf implies is that 85 percent of the faults are identified during first pass of the program and the remaining ones are identified after one diagnostic reconfiguration during the second pass. The identification time pdf is very similar for other cards as well with impulses of decreasing magnitudes at times corresponding to 1, 2, 3, and 4 passes of the SCC program. This density function for all the faults is shown in Figure 55.

The last recovery phase, system reconfiguration, is also deterministic in nature. For a given faulty module and a given system configuration, a fixed amount of time is required to replace the faulty module. Reconfiguration time pdf for all the faults is shown in Figure 56. It is seen that almost all the faulty modules are removed within 200 milliseconds.

Figure 57 shows the pdf of the total recovery time (sum of detect, identify, and reconfigure) for all the faults. The FTMP recovers from almost 95 percent of the faults within a second. Figure 58 shows the exploded view of the distribution from 0 to 1 second. If the BGU faults were excluded from the ensemble, which is reasonable to do since no self-test programs were written for it, the resulting distribution appears as shown in Figures 61 and 62. This appears almost totally identical to the pdf with BGU faults with the only exception being the average total recovery time. This is seen to drop from 1.16 to

0.549 second. In any event, the density function of the recovery time does not appear to be exponential as assumed in reliability modeling. As seen in Figure 62, it may be characterized as Gaussian density function from 0 to 1 second with an asymmetric tail going out to about 22 seconds. Whether this pdf is better or worse than the exponential pdf from the viewpoint of its impact on system reliability can only be determined through mathematical modeling. However, it is encouraging to note that a very high fraction of all incidences (about 95 percent) lie in a narrow time band around the average value. This can only have a favorable impact on the system reliability.

3.4 Actual Failures

The mean time between failures of an LRU was assumed in the reliability models to be 2,600 hours. Based on the observed failure rate during a course of 18 months of routine FTMP operation, the LRU MTBF can be estimated to be at least 10,000 hours. Over 130,000 LRU operating hours were accumulated during this time and only 12 failures were observed. Of course, this experience has been obtained in a laboratory environment which is not subject to the temperature variations and shock and vibration induced by turbulence and landings and take-offs. In that regard the laboratory environment is certainly more benign. However, the equipment was subjected to substantial power cycling much more than might be expected in the field. Also, the electronic components were going through their burn-in period during which they are known to have a higher failure rate. In fact, almost all the failures observed can be attributed to burn-in.

Two types of components accounted for almost all the failures. One was the Harris random access memory chip. There were four RAM chip failures. The second component that failed was a new 1553 LSI chip. Six of these failures were observed. In addition, two diodes in two LRU power converters failed, shorting two power buses together although they did not impact LRU or system operation.

Finally, an actual single point fault occured in one LRU that resulted in a total system failure. The faulty component was a voltage regulator in the recharging circuit for the battery that provides LRU backup power. This backup battery power is used to hold the contents of the CMOS circuitry, including the configuration control registers in the BGUs, in the absence of the primary power.

When the voltage regulator failed the output voltage of the backup power going to the BGU registers exceeded the safe high limit. This caused the LRU enable registers in both BGU cards to behave erratically and enabled the subject LRU on multiple system buses simultaneously. This, in turn, made the system bus useless leading to the system failure.

This failure mode which is basically a common failure mode of the two bus guardian units was not overlooked in the specification process. As a matter of fact, anticipating such common failure modes the FTMP design specification called for undervoltage and overvoltage protection circuits on individual BGU cards. Unfortunately, during detailed circuit level design of the bus guardian units the overvoltage protection circuitry was omitted from the design. This omission was not caught during subsequent design reviews.

CHAPTER 4

SUMMARY AND CONCLUSIONS

A total of 21,055 pin level faults were injected into the FTMP. Of these, 17418, or 83 percent, were detected. Of the 3,637 undetected faults at least 80 percent were estimated to be on unused gates and pins. A few of the remaining undetected faults were analyzed and found to belong to the 'don't care' class. Further analysis of undetected faults is required to arrive at a definitive detection coverage value. Identification and reconfiguration coverages, on the other hand, were found to be perfect for the detected faults. The system identified all detected faults correctly and successfully recovered in each case.

The total time to recover from a fault was dominated by time spent in the detection phase. Time to identify a fault and reconfigure the system was found to be deterministic and bounded, as expected. Average identification and reconfiguration times were found to be 88 and 82 milliseconds, respectively. Total recovery time averaged over all 17,418 faults was found to be 1.16 second although it would be only 549 milliseconds if BGU faults were excluded. In the absence of BGU self-test programs, which was the case here, BGU faults were solely uncovered by very low frequency routine system reconfigurations.

The distribution of the total recovery time does not appear to be exponential. However, it is encouraging to note that over 95 percent of all faults are recovered from in a second or less.

In addition to this hard data, a number of very important though intangible results were obtained as well. The hardware and software, in general, and the fault detection hardware and the fault identification and system configuration control software, in particular, performed extremely well under the stress of thousands of faults. In a sense the FTMP architecture, the hardware, and the software have been validated informally.

The test and evaluation experiments, their positive results, and the 100 percent availability of the FTMP during 13,000 hours of routine operation at the Draper Laboratory have all substantially bolstered confidence in the FTMP concept as well as its realization in hardware and software.

LIST OF REFERENCES

1. Lala, J.H., and C.J. Smith, "Performance and Economy of a Fault-Tolerant Multiprocessor," Proceedings of the National Computer Conference, Vol. 48, New York, NY, June 1979.

2. Hopkins, A.L., T.B. Smith, and J.H. Lala, "FTMP - A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," Proceedings of the IEEE, Vol. 66, No. 10, October 1978.

1 Report No	2 Government Accession No	3. Recipient's Catalog No
NASA CR-166073		

4 Title and Subtitle	5 Report Date
DEVELOPMENT AND EVALUATION OF A FAULT-TOLERANT MULTIPROCESSOR (FTMP) COMPUTER Volume III - FTMP Test and Evaluation	May 1983
	6 Performing Organization Code
7 Author(s) J. H. Lala and T. B. Smith III	8 Performing Organization Report No CSDL-R-1602
	10 Work Unit No
9 Performing Organization Name and Address The Charles Stark Draper Laboratory, Inc. 555 Technology Square Cambridge, Massachusetts 02139	11 Contract or Grant No NAS1-15336
	13 Type of Report and Period Covered Contractor Report
12 Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546	14 Sponsoring Agency Code

15 Supplementary Notes

Langley Technical Monitor: Charles W. Meissner, Jr.
Final Report

16 Abstract

This report is Volume III of a four-volume final report on the Fault-Tolerant Multiprocessor (FTMP) project. It covers in detail the experimental test and evaluation of the FTMP. Major objectives of this exercise include expanding validation envelope, building confidence in the system, revealing any weaknesses in the architectural concepts and in their execution in hardware and software, and in general, stressing the hardware and software.

To this end, pin-level faults were injected into one LRU of the FTMP and the FTMP response was measured in terms of fault detection, isolation, and recovery times. A total of 21,055 'stuck-at-0', 'stuck-at-1' and 'invert-signal' faults were injected in the CPU, memory, bus interface circuits, Bus Guardian Units, and voters and error latches. Of these, 17,418 were detected. At least 80 percent of undetected faults are estimated to be on unused pins. The multiprocessor identified all detected faults correctly and recovered successfully in each case. Total recovery time for all faults averaged a little over one second. This can be reduced to half a second by including appropriate self-tests.

17 Key Words (Suggested by Author(s))	18 Distribution Statement
Fault-Tolerance Multiprocessor Synchronous Reconfigurable Fault Injection	~~FEDD~~ Distribution Subject Category 62

19 Security Classif (of this report)	20 Security Classif (of this page)	21 No of Pages	22 Price
Unclassified	Unclassified	115	

Available: NASA's Industrial Applications Centers

End of Document