JPRS-CST-86-026

8 JULY 1986

China Report

SCIENCE AND TECHNOLOGY

SELECTIONS ON '757' VECTOR COMPUTER

FBIS FOREIGN BROADCAST INFORMATION SERVICE

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

NOTE

JPRS publications contain information primarily from foreign newspapers, periodicals and books, but also from news agency transmissions and broadcasts. Materials from foreign-language sources are translated; those from English-language sources are transcribed or reprinted, with the original phrasing and other characteristics retained.

Headlines, editorial reports, and material enclosed in brackets [] are supplied by JPRS. Processing indicators such as [Text] or [Excerpt] in the first line of each item, or following the last line of a brief, indicate how the original information was processed. Where no processing indicator is given, the information was summarized or extracted.

Unfamiliar names rendered phonetically or transliterated are enclosed in parentheses. Words or names preceded by a question mark and enclosed in parentheses were not clear in the original but have been supplied as appropriate in context. Other unattributed parenthetical notes within the body of an item originate with the source. Times within items are as given by source.

The contents of this publication in no way represent the policies, views or attitudes of the U.S. Government.

PROCUREMENT OF PUBLICATIONS

JPRS publications may be ordered from the National Technical Information Service, Springfield, Virginia 22161. In ordering, it is recommended that the JPRS number, title, date and author, if applicable, of publication be cited.

Current JPRS publications are announced in Government Reports Announcements issued semi-monthly by the National Technical Information Service, and are listed in the Monthly Catalog of U.S. Government Publications issued by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

Correspondence pertaining to matters other than procurement may be addressed to Joint Publications Research Service, 1000 North Glebe Road, Arlington, Virginia 22201.

JPRS-CST-86-026

8 JULY 1986

CHINA REPORT

SCIENCE AND TECHNOLOGY

SELECTIONS ON '757' VECTOR COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Nos 2, 5, 8, 10, 1984

[Page numbers in parentheses denote page numbers in original]

CONTENTS

CHINA'S FIRST LARGE VECTOR COMPUTER

Beijing [JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1982 pp 1-2

[Article by Wang Shuhe [3769 2885 0735], Institute of Computing Technology, Chinese Academy of Sciences (CAS): "The 10-MIPS 757 Vector Computer System Passes State Evaluation"]

[Text] The State Evaluation meeting of the "757" 10-MIPS [million instructions per second] computer system was conducted from 13-15 November 1983 in Beijing. The meeting was carried out by a State Appraisal Board authorized by the State Council's Computer and Large-Scale Integrated Circuit Leading Group. The session was attended by 120 representatives of 50 units nationwide. Comrade Fang Yi [2455 3015], State Council member and State Scientific and Technical Commission chairman, and CAS Director Comrade Lu Jiaxi [4151 0857 6932], spoke at the meeting.

The State Appraisal Board, consisting of 26 eminent Chinese computer experts and leadership personnel from the various departments concerned, worked conscientiously at the conference. It heard the research report delivered by Comrade Wu Jikang [0702 0415 1660] on behalf of the CAS Institute of Computing Technology, which developed the "757," the technical evaluation report made by Comrade Jin Yilian [6855 1837 3425] on behalf of the Technical Evaluation Group (see appendix), and reports by four subgroups of the Technical Evaluation Group. The board also studied all related documentation and technical data. Ultimately, it unanimously granted state certification for the 10-MIPS 757 computer system and held a signing ceremony.

The State appraisal certificate says the following:

After conscientious discussion, the State Appraisal Board has concluded that the technical evaluation report of the 757 machine corresponds with the actual situation and is appropriate. The board has decided to approve this report. The 757 machine is China's first exclusively Chinese-developed large vector computer. An investigation of the results has revealed that the principal technical and reliability indexes either meet or exceed the requirements of the appraisal guidelines. The State Board certification of the "757" is an important mark of China's increased level of capability in R&D of large computers.

The "757" is the fruit of self-reliance and energetic cooperation, and the result of close coordination between research, production, and user units. The development of the "757" has accumulated experience for China's program of independent development of large-scale computers. In addition it indicates that the CAS Institute of Computing Technology has an outstanding technical contingent which is a valuable national resource and which should be more effectively utilized in the future.

The State Appraisal Board concludes that large-scale computers are essential to modernization. It hopes that all departments concerned will continue their efforts in strengthening basic and developmental work in computer technology, and in planning for the coordinated development of superlarge-, large-, medium-sized, mini- and micro-computers for making greater contributions to developing China's work in the computer field and to the four modernizations.

Appendix: The Technical Appraisal Report on the "757" 10-MIPS Computer System [Excerpts]

The "757" is a large computer system which was independently designed and experimentally developed by China. The objective of its development was to solve large-scale scientific and engineering problems arising in China's economic development and scientific research. The CAS Institute of Computing Technology, the principal unit responsible, conducted the "757" research and development over a period of several years with the help of cooperating units, using China's own resources.

The Technical Evaluation Group conducted the technical evaluation of the "757" from 3 August to 12 November 1983. Its conclusions are stated below.

I. Hardware

The 757's hardware system includes a mainframe, a peripheral processor and various peripheral devices. The Evaluation Group has tested the hardware. It has carried out frequency-shift, power-supply variation and noise immunity tests on the various components, as well as testing the speed of the machine. It has confirmed that the 757's operating speed, the performance of all components of the "757" machine and all peripherals, and the range of stable operation of the entire machine meet the required indicators of the evaluation guidelines.

II. Software

The 757's software system includes an operating system, a vector FORTRAN compiler, a mainframe assembler, a peripheral processor, internal function subroutines, a basic graphics package, a diagnostic program and a double-calculation program.

The Evaluation Group investigated the operating system's capabilities and reliability. It examined the vector FORTRAN compiler's correctness and error reporting capability and analyzed its efficiency. In addition, an

analysis was made of the assembler, the basic graphics package and the internal function subroutines, and the diagnostic and double-calculation programs were tested. It was concluded that the 757's software system meets the requirements of the evaluation guidelines.

III. Problem Solving Tested

In order to investigate the actual computational capability and reliability of the "757," between May and October 1983 we ran on the mainframe machine 30 assembler problems, 10 vector FORTRAN problems, and 25 scalar FORTRAN problems. These problems represented a wide range of users. The runs indicated that the computation results were correct and the precision met the requirements.

IV. System Reliability

From 24 October to 8 November 1983, we carried out a 15-consecutive day reliability test of the "757," including multiprogramming operation. During the testing period, we ran test programs with known results designed particularly for checking the mainframe and the peripheral processor. Statistics indicated the following:

$$\text{System mean time between failures} = \frac{\text{total time of normal operation}}{\text{number of failures + number of jitters + 1}} = 120 \text{ hours}$$

$$\text{System mean time to failure} = \frac{\text{total time of normal operation}}{\text{number of failures + 1}} = 120 \text{ hours}$$

$$\text{System availability} = \frac{\text{total time of normal operation}}{\text{total time of normal operation + total maintenance time}} =$$

99.8 percent

The above results all exceeded the requirements of the evaluation guidelines.

The Technical Evaluation Group discussed the results of the evaluations and unanimously concluded that the "757" machine is the first large exclusively Chinese-developed vector computer. It was built on the basis of China's then current technical capabilities and with Chinese-made components and equipment. In system design, the concepts of vector crossbar in-processing [zongheng jiagong [4912 2897 0502 1562]] and multiple [duo [1122]] vector accumulator were independently proposed. In logic design, pipeline and overlap techniques were adopted. Vector operations reached 10 million per second, and scalar operations reached 2.8 million per second. The peripheral devices are relatively complete. The system software is also fairly complete. A FORTRAN-77 compiler developed earlier in China and its vector functions were expanded. The operating system has a multiprogramming capability. The system also makes use of checking, correcting, double-calculating and diagnostic techniques, has effectively increased machine reliability, availability, and maintainability. Tests indicate that the machine is stable and reliable and that its principal technical and

reliability indexes either meet or exceed the requirements of the evaluation guidelines. The machine has excellent performance capabilities.

The development of the "757" machine was based on domestic technical capabilities and made use of domestically produced materials and equipment, which stimulated the development of China's basic computer components and processes. Computer-aided design (CAD) was used during the R&D period that not only speeded up the engineering process, but also promoted the development of domestic CAD technology.

The "757" machine is the fruit of self-reliance and large-scale cooperation and the result of close coordination and common effort between development, production, and user units. Overcoming difficult problems related to new technologies and processes, more than 80 units of the CAS Institute of Computing Technology and more than 30 departments and localities nationwide cooperated on a large scale. The successful development of the 10-MIPS "757" system has provided experience that is usable in China's independent large computer development project.

The Evaluation Group has also concluded that in the future we must further develop software suited to the characteristics of the "757" and energetically pursue computation and algorithm research so as to make the fullest use of the 757's high-speed problem-solving capabilities.

8480/9365
CSO: 4008/199

INTRODUCTION TO 757 VECTOR COMPUTER SYSTEM

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 3-6

[Article by Wu Jikang [0702 0415 1660], Zhao Renchang [6392 0088 2490], and Wang Zhenshan [3769 2182 1472], Institute of Computing Technology, Chinese Academy of Sciences (CAS)]

[Text] In order to solve large-scale scientific and engineering problems arising in China's four modernizations, the CAS Institute of Computing Technology (shown in Figure 1) recently developed China's first 10-MIPS large computer, the "757" machine (see Figure 2). Its development has successfully raised the level of China's computer research and development activities and has promoted the development of computer science and technology in China.

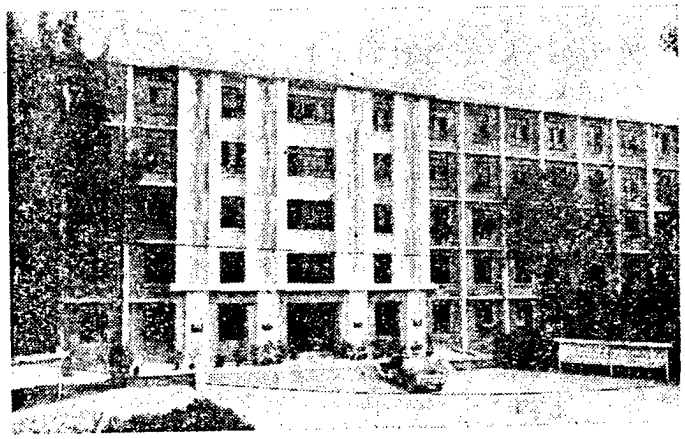


Figure 1. One of the Research Buildings of the CAS Institute of Computing Technology

The system hardware of the "757" consists of a vector processor (mainframe) and a peripheral processor (see Figures 3 and 4).

The general design approach in the mainframe is that of a crossbar in-processing [zongheng jiagong [4912 2897 0502 1562]] vector machine. Its characteristics are based on China's national situation, the multiple [duo [1122]] vector accumulator concept was introduced into its design,

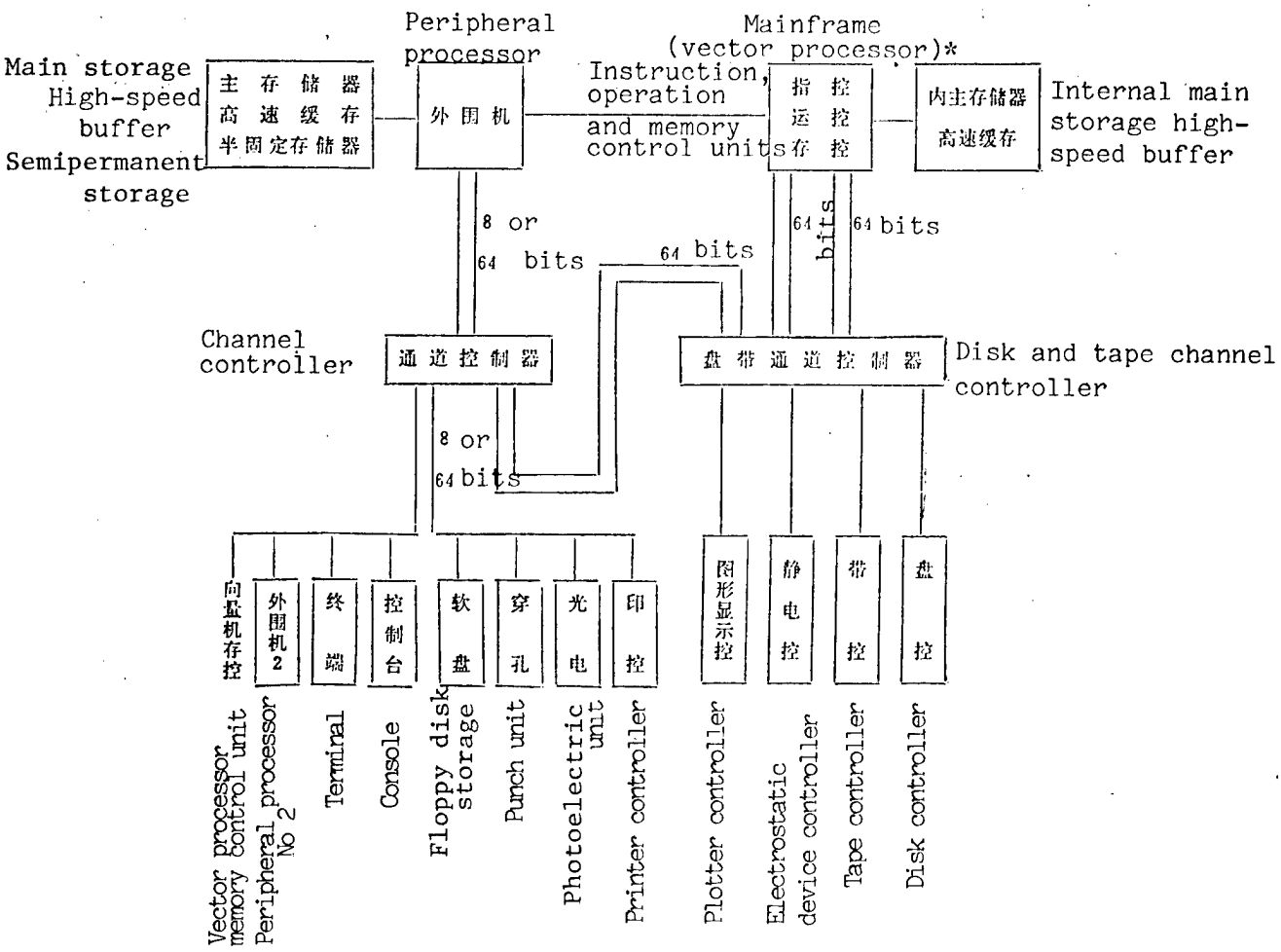


Figure 2. The "757" System Flowchart
(*The mainframe also includes the ALU)

Figure 3. Vector Processor of the "757" Large Computer (Mainframe)

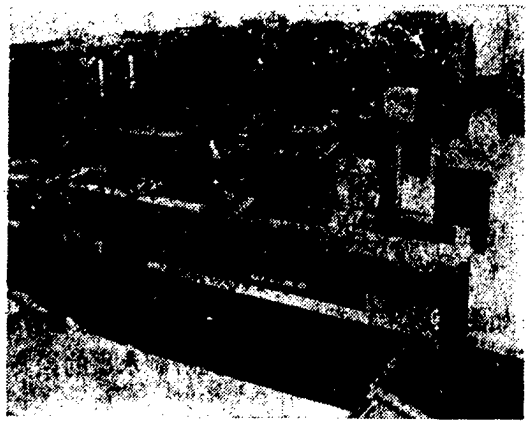


Figure 4. Peripheral Devices of the "757" Large Computer

making it possible to expand the capability of the main storage to three to four times that of conventional high-speed pipelined machines with equal efficiency. Architecturally, it employs three main control elements, namely the instruction control unit, the operation unit and the memory control unit. It also utilizes a high degree of overlap between the ALU [arithmetic-logic unit] and memory components. The 16 main memory units are parallel-interleaved. The power of the instruction set is quite considerable. Thus, it was possible to convert a conventionally-designed 2-MIPS computer's low-speed components and modules to build the large 10-MIPS high-speed "757" machine.

The hardware system includes a vector processor, a peripheral processor, and peripheral devices. The vector processor conducts vector processing in a single-pipeline structure, using the crossbar in-process [4912 2897 0502 1562] method. This is rather efficient for large-scale scientific and engineering computations involving mainly parallel calculations; the average operating speed is 10 MIPS. The average speed for scalar computations is 2.8 MIPS. The vector word length and instruction word length are both 64 bits. Data types include full word floating point number, half word floating point number, double word floating point number, signed integers, and unsigned integers (including 1-, 2-, and 4-byte integers). There are a total of 107 instructions, of which 97 are user-accessible, while the other 10 are for system use.

The vector processor consists of an instruction control unit, an operation control unit, a memory control unit and a main memory system. The three controllers use Chinese-made ECL [emitter-coupled logic] medium-scale and small-scale integrated circuits [MSI, SSI], with an average level of delay time $t_y \leqslant 4$ ns and a clock frequency of 8.2 MHz. The superhigh-speed buffer memory uses Chinese-made LSI [large-scale integration] circuits and has an access cycle of 100 ns and a read time of 30 ns. The main storage has a capacity of 520,000 words 72 bits long (including an 8-bit check code). A cycle is 1.5 μs, and the fetch time is 800 ns. The system consists of 16 individual units (plus 2 backup units), and its maximum access speed is 8.2 million words per second in modulo-16 operation. Two hot-standby units allow operator-initiated or automatic switchover. It can also be switched over partially. The system in degraded operation can function in either modulo-8 plus modulo-4 or single modulo-8.

The peripheral processor is a medium-size computer. Its word length is 64 bits and it operates in the fixed-point mode with a main clock frequency of 2.5 MHz. Its average operating speed is 500,000 operations per second. It runs primarily operating system and compiler system programs. It consists of a central processor, main storage, semipermanent storage (with capacity of 64K and a word length of 65 bits) and a channel controller. It has an internal memory of 64K with a word length of 64 bits plus an 8-bit check code. The channel controller has 32 channels. There are 45 peripheral devices of 9 types. They include magnetic disk drives, magnetic tape drives, printers, photoelectric input units, punch output units, electrostatic printers, keyboard displays, graphic displays, and floppy disk inputs and outputs.

The peripheral processor handles all language compilations, input and output data processing, management of the peripheral devices, and most of the system management. It relieves the mainframe of a large amount of time-consuming, low-efficiency tasks so that it can concentrate on running user programs, thus assuring that the high-speed operating capabilities of the vector processor will be fully utilized and ensuring the problem-solving efficiency of the entire system.

In order to increase system reliability, computation speed, and efficiency, the "757" machine not only has pooled the successful experience acquired in past development of many large- and medium-sized computers, but in addition referenced certain then-advanced international technologies as well, so that the machine has many unique characteristics in both hardware and software engineering.

The logic design of the mainframe's instruction set, arithmetic unit, and memory control components all have used some new algorithms and control methods. For example, the instruction control unit uses a control method combining beat and overlap, as well as high-speed instruction buffer technique; the ALU uses iterative division, multidigit parallel multiplication, and direct-code [i.e. not complement] addition method; the memory control unit uses the Hamming code error correction technique and modulo-16 cross access and dual backup memory unit design methods. These features make the design of the three control units considerably more sophisticated than in previous models and are a major factor enabling the machine to operate reliably at 10 MIPS.

The main memory circuit design of the vector processor has been improved by lowering the utility voltage and decreasing component power consumption. In addition, the Institute personnel visited the plants and worked with them to improve the quality of components, so that the reliability indexes of individual memory units have been improved markedly. The average length of stable operation has been increased from 100 hours to more than 500 hours (excluding Hamming code checks).

The main work on the peripheral processor was to focus on improving reliability and convenience of operation. Because painstaking work was done on logic design and engineering realization, it was the first section of the system on which work was completed. In the first test, the longest period of fault-free operation was 418 hours.

The "757" machine has a total of 45 peripheral devices of 9 types. The total capacity of its magnetic disk storage is 16 million words (a total of 8 units at 2 million words per unit). Some of the disk packs are domestically produced (shown in Figure 5). It is a gratifying step to be able to develop these from scratch, thereby laying the groundwork for future Chinese development of large-capacity disk storage. The magnetic tape storage uses the advanced international GCR [group-coded recording] method. They are superior to existing Chinese-made tape storage in both recording density and error-correcting capabilities. Recording

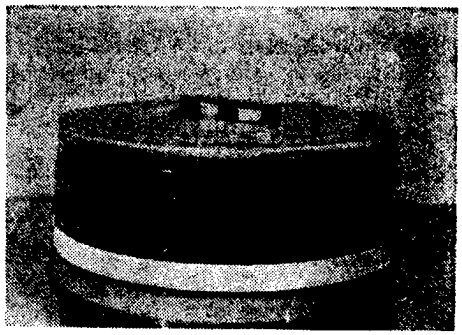


Figure 5. Removable Disk Pack

density has been increased from 500 bits per inch to 2,500 bits per inch. Others, such as the use of (2,048 x 2,048) bits per frame high-density electrostatic printout device and a microprogram-controlled wide line printer, as well as a color graph plotting output unit and floppy disk input-output units, make the "757" machine with peripheral devices superior to those of previous computers. This is a gratifying change for Chinese-made computer peripherals, which have always been a weak link.

The quality of the power supply affects the reliability of the entire computer. The "757" uses a zero [wugong [3541 1562]] frequency input transformer, which greatly decreases the dimensions of the power supply, saves energy, and improves efficiency. The power source to the mainframe's three control units operating in parallel is realized, forming a redundant system, which increases the reliability of the power supply system.

As for mechanical design, the vector processor is laid out in a circular configuration in order to shorten the amount of wiring and increase the speed. In addition, it uses multilayer printed circuit boards, wire wrap, and other new technologies. Standard cabinets are used throughout the machine. A static-pressure parallel short-conduit ventilating system is used, which is simple and easy to operate and gives excellent results.

The 757's software system includes an operating system, vector FORTRAN and a compiler, a mainfrane assembly language and assembler, internal function subroutine, a basic graphics package (BGP), and a peripheral processor assembly language and assembler, as well as a diagnostic program, a double-calculation program, etc. The system software totals nearly 20,000 instructions.

The "757" system software not only has considered the entire system's high degree of pipelined overlap and vector operation, but also every effort is made to assure reliability and ease of operation for the user.

The operating system was initiated primarily for reliability and high efficiency and sets up preferential tasks; as such its scope is in the middle. It is a multiprocessing batch-oriented operating system. I allows 31

programs to be put in the backup state and 16 programs in the executing state; 9 programs can be run in the vector processor and 2 in the peripheral processor [?]. The system can take protective measures when errors arise during operation; these include an occasional jitter in double-calculations, coordinate double-calculation diagnosis, and continuation of operation from breakpoint. The system has a monitor and a timer (compatible with the M-170), billing and other accounting functions; document security is assured through a variety of passwords.

The "757" vector FORTRAN language is the binding characteristic of the vector processor. Its design is based on the new international standard FORTRAN-77. It has been suitably expanded in terms of data types, vector components and its operational phase. In addition, the "757" vector FORTRAN is upwardly compatible with FORTRAN 66. Because a necessary expansion has been carried out, programs written in vector FORTRAN can now make better use of parallel vector processing to realize high-speed calculations.

The system software also provides 18 basic internal functions and 92 standard subroutines. It includes three types of numbers--single precision, double precision, and complex numbers.

The merits of the basic graphics package are device-independence and ease of use. In connection with the 757 mission, software development involved program design tools, R&D of program structure nd methods, and establishment of a primarily processing-oriented tool language, EML, which was used to write all of the vector FORTRAN assembly programs.

In order to increase the efficiency and reliability of the "757," a special mainframe diagnostic system was designed. It uses the peripheral processor to carry out diagnostics for the mainframe. The diagnostic system has error detection, alarm and retry capabilities. Diagnosis is performed automatically, and for transient faults the system automatically tests for reexecutibility by trying to reexecute as many as seven times. The extent of overlay varies in the different component units but is generally 50 to 90 percent.

In the case of permament faults, the system uses both automatic and manual diagnosis. The automatic card identification feature can generally determine the location of a fault to within a few cards or even to one specific card. The vector processor's memory can automatically correct faults, identify malfunctioning boards or perform switchovers, depending on the nature of the fault.

The "757" is Chinese-designed and based on Chinese technical capabilities, and uses domestically-produced components (the system uses LSI, MSI, and SSI integrated circuits and has over 300,000 parts and components of more than 40 types). As described above, both the entire system and the hardware and software separately were subjected to painstaking design and stringent engineering realization. In order to assure high quality and to speed up progress on the project, we used computer-aided design (CAD) virtually throughout. We used earlier Chinese-developed computers to design

this large 10-MIPS computer. In order to test the stability and reliability of the system, before it was submitted for state evaluation we performed 15 consecutive days of continuous reliability testing. The 20 test problems which were used were rather representative and fairly difficult. The results indicated that processing the test problems had yielded correct results; the average period of stable operation of the system (including mainframe, peripheral processor and peripheral devices) was found to be 120 hours, the system availability was 99.8 percent, and the longest single period between failures was 205 hours, 40 minutes. By domestic standards these figures are all advanced achievements for large general-purpose computers.

The successful development of the 757 was the result of self-reliance and large-scale coordination. Thirty departments and localities and more than 80 units nationwide took part in the process, and all made a maximum effort. The participation of this large number of departments and units indicates the immense scale of the "757" development process. This gives us the gratifying realization that the consequences of the successful development of the "757" are not only embodied in the computer itself, but also are present in a series of new processes, technologies, and products that grew out of the project. Thus a certain stimulus has been given to the improvement of China's computer development standards and to the further development of computer science and technology.

8480/9365
CSO: 4008/199

SYSTEM ARCHITECTURE OF 757 VECTOR COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 7-16

[Article by Yang Shufan [2799 2885 5400], Institute of Computing Technology, Chinese Academy of Sciences (CAS)]

[Text] The system architecture, main performance characteristics, technical specifications and design characteristics of the "757" vector machine are surveyed. In hardware, attention is devoted to the method of making thorough use of the vector machine's lengthwise and crosswise processing capabilities and overcoming collisions, the fatal problem in pipelined machines; to the method of dealing with speed allocation among the three control units of the vector machine; and particularly to the method of handling supply-demand conflicts between the high-speed central processor and low-speed magnetic core storage.

I. Overview

The 757 computer system is a large general-purpose computer oriented to vector computation. It consists of a vector machine, a peripheral processor and various peripheral devices, as well as software. The vector machine has a parallel overlapped single-pipeline structure and is constructed entirely of Chinese-made components. The present article focuses on the hardware structure of the vector machine.

1. Main Characteristics of the Vector Machine

Word length: 64 bits (including operand and instruction). Main memory capacity: 512 K words, word length 64 bits, with 8 additional check bits. The operand fetch address can be refined to the byte level. Main clock frequency: 8.2 MHz. Speed: in problems suited to parallel computation the machine operates at high or medium efficiency and the average computation speed is about 10 MIPS, while for scalar operations the system is in a low-efficiency state and the average speed is 2.8 MIPS.

2. Technical Specifications

A. Components: The machine uses exclusively Chinese-made medium- and small-scale integrated circuits of the ECL [emitter-coupled logic] type. The circuit series includes 10 production types: besides the 5 types of gate circuits, 2 model D flip-flops, a 4-bit half adder, an 8-bit shift register, and a semiconductor memory unit.

Speed characteristics: Gates and half-adders, average gate delay $t_{pd} \leqslant 4$ ns; flip-flops, $t_{pd} \leqslant 6$ ns; shift register, $t_{pd} \leqslant 10$ ns. Semiconductor memory: cycle time $T \leqslant 100$ ns, readout time, $t_d \leqslant 40$ ns; average failure rate: 3×10^{-8} (statistics for January-August 1983).

B. Main Memory: Uses magnetic core storage. Access cycle $T \leqslant 1{,}500$ ns, read time $t_d \leqslant 800$ ns. Each memory unit has a capacity of 32K words with a word length of 64 + 8 bits. The main memory system consists of 18 units, of which 2 are on warm standby and can switch over automatically. When necessary the system can automatically cut-off a part of the memory unit and go into degraded operation. The system can operate in modulo-16, modulo-8 + modulo-4, and modulo-8 form.

C. Pulse System: The system uses a single-clock synchronous pulse system with a principal frequency of 8.2 MHz and a pulse width of 50 to 60 ns.

D. Interconnection Technique: Multilevel printed circuits (including cards and boards) and twisted pairlines. System impedance 90 ohms.

E. Vector Machine Layout: (Other than the main memory) circular arrangement, consisting of 11 racks, each with 10 printed circuit boards; a total of 107 circuit boards and 1,119 cards with about 45,000 integrated circuits are used.

F. Cooling Method: Air cooled. Incoming air temperature 17 ± 1°C, relative humidity, 40 to 60 percent. Module surface temperature < 55°C, maximum temperature difference between modules < 30°C.

3. System Reliability Provisions

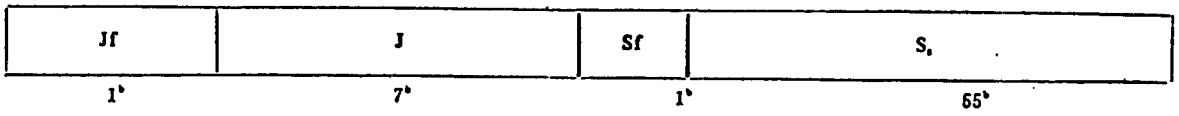
The mainframe's three control units use parity checks, backup equipment diagnostics, and a double-calculation system. The double-calculation can be made at the instruction execution level or the register level. Fault diagnosis and location is accurate to within 1-3 cards; the main memory uses an odd-weighted code error detection system, while the two external backup units can perform operator-initiated or automatic switchovers. It is estimated that adoption of these measures can increase the relative reliability of main memory by more than 3.5 times. The system can also cut off a unit, and in the degraded mode it can function in modulo-8 + modulo 4 or simple modulo-8 form. These capabilities are implemented by the memory control unit.

II. Data and Instruction Formats

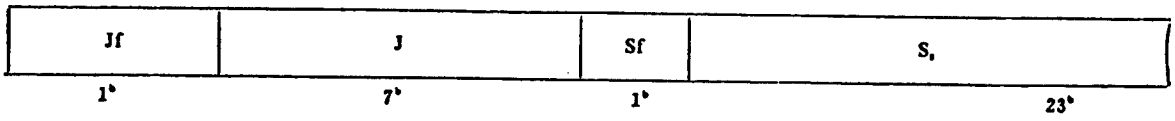
1. Data Formats

Floating point numbers: It consists of the characteristic sign (Jf), the characteristic code (J), the number sign (Sf), and the mantissa (Ss). The characteristic is expressed in complement form and the mantissa in basic form, and machine language zero is expressed by "0". There are three types of floating point:

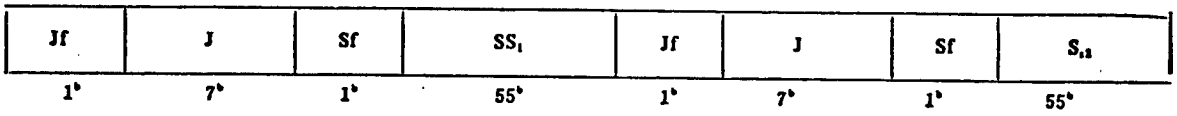
1) full word floating point (64 bits), occupies one memory cell.



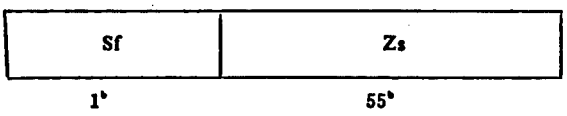
2) half word floating point (32 bits), occupies half a memory cell.



3) double word floating point (128 bits), occupies two memory cells. The two cells have the same characteristic and number sign. S_{S1} is the high-order part of the mantissa, and S_{S2} the low-order part.

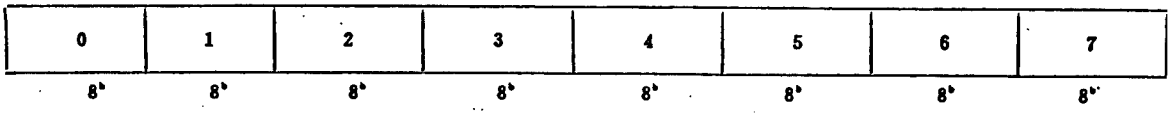


Signed integers: These are in direct code, consisting of 56 bits including sign.

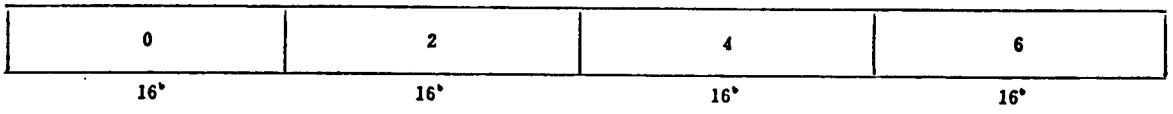


Unsigned integers: There are three types, stored in main memory as follows:

1) one-byte integers (8 bits), with each memory cell containing eight integers;



2) two-byte integers (16 bits), with each memory cell containing 4 integers; and

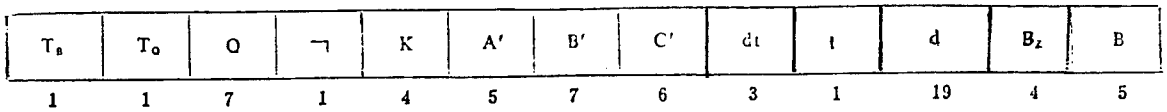


3) four-byte integers (32 bits), with each cell containing 2 integers.

In addition there are 1-, 2-, and 4-byte and full and double-word length codes, number codes, and bits.

2. Instruction format: There are 97 user-accessible instructions and 10 system instructions. The user instructions are of two types:

Operation instructions (77), with 4-address format:



These include: Q (opcode, 7 bits); T_B (vector-scalar flag, when T_B = 1 indicates a scalar instruction); T_Q (operand fetch flag, when T_Q = 1 indicates fetch from internal storage, and when T_Q = 0 indicates transfer to main storage); A', B', and C' (indicate vector-scalar or vector-scalar accumulator and specify the source operand and target accumulator address); K (indicates control vector or control scalar); ¬ (indicates the use of the base-minus-one's complement of (K) for control).

If we define: $(X'){::=}(D')\Rightarrow A'|(D')$, $Z'{::=}(C')\Rightarrow D'|D'$, then the operation instruction can be described as follows: $(X')Q(B')\Rightarrow Z'$

Addressing Mode: A main storage address D' is described by the combination T_B, dt, d, !, BZ, and B; the address may refer to a byte. When d = 0, $b = b_0$, internal storage is not accessed and the instruction is an accumulator-type instruction.

Element dt is the byte flag, which indicates the type of data to be fetched; ! is the successor symbol; d is the formal address (19 bits); B is the index and refers to b_{0-31}, with a word length of (B) = 22 bits; BZ is the increment, and when BZ_{0-11} is referred to, (BZi) is the equidistant vector increment, when BZ_{12-15} is referred to, this refers to the indirect address register ($\vec{q}_{0-3}$ or q_{0-3}) where (q) is 22 bits long.

For equidistantly stored data:

$$\begin{aligned} \text{scalar } D' &= d + (B) \\ \text{vector } D_0 &= D' + 0\cdot(BZ) \\ D_1 &= D' + 1\cdot(BZ) \\ &\vdots \\ D_{\ell-1} &= D' + (\ell-1)\cdot(BZ) \end{aligned}$$

For indirect addressing:

$$D' = d + (B)$$
$$\text{scalar } D = D' + (qi)$$
$$\text{vector } \vec{D} = D + (\vec{q}i)$$
$$1 \leqslant \ell \leqslant 16$$

Once the logical address D is found, the real address is determined from the page table.

Control instructions (20), including jumps, loops, index operations, index transfers and the like. The instruction format is:

T_1	T_2	Q		m_1	m_2	N		d	BZ	B
1	1	7	3	7	7	6	4	19	4	5

Here T_1 is the instruction dispatch flag. For jump instructions, T_1 controls whether or not the subsequent instruction should continue to be dispatched to the next instruction register (ZH). T_2 is the locator flag, which is only used in switch instructions. In loop segments with fewer than 64 instructions, T_2 can be used to control loop execution in ZH. The reason for providing T_1 and T_2 is to minimize transfers from internal memory and to increase the efficiency of ZH. Q is the opcode (m_1, m_2, and N indicate B, BZ, KB_{0-15}, JN, and J_{Dt}, the registers involved in instruction control, in order to carry out such operations as jumps, index operation control, and index transfer.

III. The 757's System Architecture and Features

Architecture of the "757" Machine

The 757's vector machine, peripheral processor, and peripheral devices are shown in Figure 1.

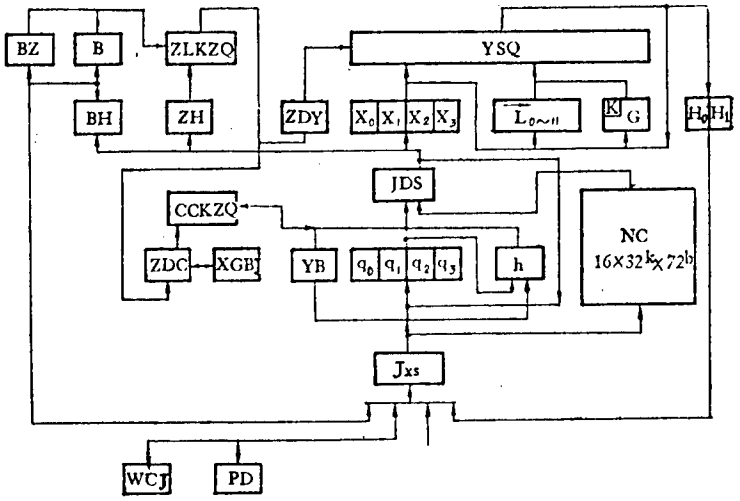


Figure 1. Block Diagram of the 757 Architecture

The peripheral processor has a word length of 64 bits and carries out floating point operations. It has a main clock speed of 2.5 MHz and an average speed of 500,000 Hz. Its internal memory capacity is 64,000 words. The semifixed storage capacity is 64,000 words. The peripheral machine runs primarily system programs, while the mainframe (vector machine) runs primarily system applications programs. There are two channels connecting the mainframe and peripheral processor: one is a communication channel, using the interrupt method, by which they exchange control information; the other is a direct batch transmission channel in the time sharing mode. Information transfer between the mainframe and the disks, tapes, electrostatic printers and graph plotters may also be in the time-sharing mode. The disk resources are shared by the main and auxiliary machines. All peripheral devices operate under the control of the peripheral processor.

The mainframe (vector machine) has a single pipeline structure and basically belongs to the register-register class. The mainframe consists of the instruction control unit (ZK), the operation control unit (YK), the memory control unit (CK), and main memory (NC). These are described below:

1. The Instruction Control Unit (ZK): The ZK consists of the instruction buffer register (ZH) (4 x 16 x 64 bits), index registers B (32 x 22 bits), increment registers 13Z (32 x 22 bits), the address ALU, general purpose registers, and an interrupt processor. Its functions are to analyze instructions, execute operation or access instructions, then transfer them to YK or CK; to execute control instructions; and to respond to interrupts.

2. The Operation Control Unit (YK): This consists of an ALU, accumulators (including vector accumulators L_{0-11} and scalar accumulators L_{0-11}) with capacities of 12 x 16 x (64 + 8) bits, high-speed scalar registers G (32 x (64 + 8) bits), look-ahead operand fetch stack X_{0-3} (4 x 16 x 72 bits), look-behind operand store stack H_{0-1} (2 x 16 x 65 bits), operation control unit instruction stack ZDY (16 x 75 bits) and the requisite control circuitry; it carries out arithmetical and logical operations.

3. The Memory Control Unit (CK): The memory control unit consists of the storage control stack ZDC (16 x 87 bits), look-behind wait station JH_{0-1} (2 x 67 bits), a collision processor, auxiliary machine interface buffer register WH (16 x 65 bits), 2 sets of disk and tape interface buffer registers (2 x 2 x 16 x 65 bits), indirect address register q_{0-3} (4 x 16 x 23 bits), compress and restore vector registers (2 x 16 bits), page table registers (128 x 14 bits), an address adder, the main storage checking and correction unit, and memory unit switchover and cut-off circuitry.

It handles collisions, time-shared queuing, address processing, page mapping, memory protection, data compression and restoration, byte control, memory correction, cut-off units, switchovers, etc.

4. Main Storage (NC): The main storage consists of 18 core storage units (2 of which are backup units). Each unit has a capacity of 32K x 72 bits and an access cycle of 1,500 ns; storage operation can be modulo 16, 8 + 4, or 8.

IV. Architectural Characteristics of the 757 Vector Machine

The basic characteristic of the 757 vector machine is that it introduces vector computations in a single-pipeline architecture and uses the lengthwise and crosswise method of vector processing. This decreases the amount of instruction handling tenfold, greatly decreases the number of conditional jumps, greatly reduces the number of collisions between operations, decreases memory access requirements, and makes memory access more uniform, thus fundamentally improving pipelining effectiveness.

In order to take full advantage of parallel operation and pipelining in the 757, not only will the user have to make a continuing effort to develop parallel-oriented algorithms and write suitable programs, but in addition it will be necessary to design hardware structures suited to computers of this type. Below we describe the system architecture of the 757, primarily in terms of hardware structure design.

1. Use of a Mainframe-and-Auxiliary Computer System With Distribution of Capabilities: When solving problems in a computer system, not only applications programs, but also system programs (including the operating system, the compiler system and the like) must be run. These two types of programs differ greatly in terms of program structure, data structure, relative importance of different types of instructions in them, and operating environment. The systems programs primarily perform fixed-point scalar operations, have a high proportion of conditional jumps, and are subject to frequent interruptions, so that programs and operating environment of this type are not suited to operation on a highly overlapped, primarily vector-oriented pipelined machine. This is why the two-machine architecture is used in the 757. The auxiliary machine runs primarily system programs, while the mainframe runs primarily applications programs. This utilizes the strong points of each machine and makes thorough use of the characteristics of the vector machine.

2. The vector machine has a wide range of vector processing methods, which expand the range of vector computations and thus the area in which the machine can be used with high to medium efficiency.

(1) The operation instruction vector and scalar instruction formats within the same instruction format; a four-address format and a multi-accumulator architecture are used to decrease the number of both memory accesses and auxiliary instructions.

(2) For convenience in writing programs for lengthwise and crosswise vector processing, the instructions include vector loop switching instructions, vector subgroup loop switching instructions, and other instructions geared to vector processing.

(3) The control vectors $\vec{K}$, $\vec{h}$, and $\vec{q}$ are provided. The operation control vector $\vec{K}$ is used to control whether or not an operation is executed, which can greatly decrease branching. The compression and restoration vector $\vec{h}$

controls main memory read and write. In the processing of sparse matrices it can greatly decrease the number of memory cells used and speed up memory access. The indirect address control vector $\vec{q}$ can form a group of noncontinuous or random address strings, which is helpful in processing sparse matrices.

(4) The spacing of data stored in memory can be artibrarily chosen, affording high efficiency in alternation of direction in the processing of multidimensional vectors.

3. Highly Parallel, Overlapped Pipeline Architecture: In executing programs, the vector machine uses a 16-segment lengthwise-crosswise processing technique for vectors. All major units of the machine operate in a completely overlapped, parallel state, and also in a highly pipelined fashion. For example, the ALU can produce one floating-point addition result per clock unit, while the storage controller can read or write one datum in each clock unit. The instruction and data flows in the machine are shown in Figure 2.

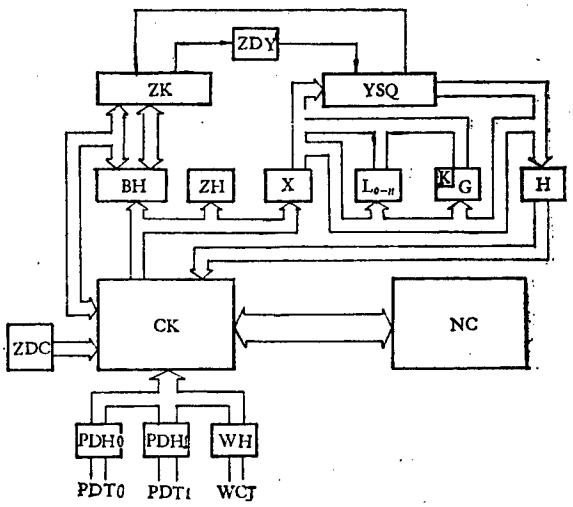


Figure 2. Information Flows in the Vector Machine

Pipelining is an effective method of high-speed operation; but in order to make full use of its capabilities, not only pipelines within all individual units, but also the parallel overlapped pipelines between the units, must be kept moving freely. Tests on seven different problem types show that sequential operating states (i.e. with the three controllers pipelined, but with ZK, YK, and CK operating serially) the machine as a whole is slower than with parallel overlap by a ratio of 1 to 1.38. The main defect influencing parallel overlapping of the three controllers is collisions. The smoothness of pipelined operation is seriously affected by collisions in conditional jumps, collisions in the processing of memory data into internal store addresses by the ALU, operation collisions, index collisions, fetch and store collisions, and memory unit collisions, as well as the collisions produced by lack of speed matching between the various machine units and between the three controllers and memory. These problems are dealt with in the following main ways.

(1) Provision of the operation control vector $\vec{K}$, which reduces conditional jumps.

(2) Provision of the address buffer register group BH_{0-3} (4 x 16 x 23 bits), which makes possible batch processing of data from memory into addresses by the ALU; these are transferred in a group to BH, from which the instruction controller accesses the needed information. There are four BH registers, which greatly decreases interference with the instruction stream.

(3) Processing of the object program by the peripheral processor: The vector machine does not process instructions, which alleviates collision between instruction fetch and data transfer channels.

(4) Use of the operating system to handle collisions between the peripheral processor channels, disk and tape channels, and mainframe internal memory access channels.

(5) Data fetch and store collisions: In the vector machine, the main information flow is data flow, accounting for the great majority of overall information flow. When operations are transferred from main memory to the operation controller via the storage controller and the processing results are transferred back to storage controller, the overall path is very long. If data fetching and storage are carried out in a program-specified sequence, the speed of the main machine is seriously affected. Therefore collision discrimination must be carried out between all fetch and store address spaces; where there is no conflict for address spaces, the data can be prefetched. Tests on the seven problem types indicate that this prefetch processing increases overall machine speed by 63 percent compared with processing in program sequence.

4. Speed Matching of the Three Control Units. Because the mainframe is primarily vector-oriented, when executing vector instructions the amount of work by the instruction control unit is decreased to $1/10 \sim 1/20$; according to analysis and system simulation, for problems involving primarily parallel processing the effective operating time of the instruction control unit is only 12.5 percent. But in order to provide for problems involving principally scalar computations or serial computations, the instruction controller is designed to process one instruction in an average of 3 to 4 beats, so that it can process from 2.5 to 3 million instructions per second. Therefore increasing the data transfer speed of YSQ and storage as much as possible and matching their speeds as closely as possible constitute the key to increasing overall speed. The memory is designed for a maximum flow rate of 1 byte access per beat while the ALU is designed for a speed of one operation per beat (with the exception of multiplication, division and certain special instructions). For example, comparison of characteristics and normalization in floating point addition both require one beat and partial operations are eliminated, which makes the pipelines more uniform. YSQ has a multibit fast multiplication algorithm and iterative division, which increase its processing speed. In order to match the speeds of the three controllers, their interfaces are provided with buffer stacks, which, in brief, are designed as follows:

(1) In order to make the data flows uniform, interference with main storage for instruction fetch has been minimized, and the instruction controller is provided with instruction buffer registers ZH (4 x 16 x 64): more than 96 percent of loop segments can be executed in ZH. Instructions can be transferred out of memory in groups with high efficiency (16 at a time), radically decreasing the frequency of instruction fetches from the main store. ZH is divided into four components and operates modulo 4. Instruction processing and instruction transfers can be entirely overlapped. Tests of the seven problem types indicate that without ZH, the speed of the main machine would be decreased by a ratio of 1 to 1.15.

(2) In order to increase the parallelism of ZK, YK, and CK, unit YK and the storage controller are provided with instruction stacks ZDY and ZDC (each with a capacity of 16 instructions).

(3) In order to keep the data flow uniform and to overlap the operation of all machine units and memory accesses, the following are provided:

--look-ahead fetch stacks X_{0-3} (4 x 16 x 72 bits);

--look-behind transfer stacks H_{0-1} (2 x 16 x 65 bits);

--index buffer stacks BH_{0-3} (4 x 16 x 23 bits);

--a peripheral processor channel data buffer stack WH (1 x 16 x 65 bits); and

--disk and tape channel data buffer stacks PH_{0-1} (2 x 2 x 16 x 65 bits).

5. Speed Matching Between Main Storage and Central Processor. The mainframe's three control units use high-speed ECL circuitry and, on average, process data at a speed of one operation per beat (approximately 100 ns), while owing to current technological limitations, the main storage uses low-speed magnetic core storage and has an access cycle of 1,500 ns. This is obviously a major conflict, and represents the key factor affecting overall machine speed. The following steps were taken to resolve it.

(1) Decreasing information flow to and from the main store by decreasing the system's memory access requirements:

a. Because lengthwise-crosswise processing of vectors is performed, it becomes possible to use multiple accumulators; the instruction controller is provided with 12 vector (or scalar) accumulators (Z_{0-11}) and a high-speed scalar buffer memory G (32 words). In this way, intermediate results and repeatedly used contents can be transferred to $\vec{L}$ and G, respectively, which considerably decreases the number of memory accesses. This decreases memory accesses by two-thirds to three-fourths or even more compared with a purely lengthwise processing machine (such as the STAR-100).

b. In order to decrease demands on memory and avoid a great decrease in speed, the 757 vector machine does not use the virtual storage dispatching method, but instead uses batch pretransfer. Because the main storage has

sufficiently large capacity and the system uses simple multiprogramming geared to one large problem, this dispatching method assures high speed and greatly simplifies the logic structure of both software and hardware.

c. Because data can be fetched and stored under the control of the compression and restoration vector $\vec{h}$, and the data can be fetched in bytewise fashion, the amount of memory space used for sparse matrices and byte operations is decreased and the number of accesses to main storage is greatly reduced, thus increasing access speed.

d. Provision of the instruction buffer registers ZH decreases the number of main storage accesses for memory fetch to 1 percent or less.

(2) Increasing the access speed of the memory system: Even with the measures described above, the system still requires a nearly 1-to-1 memory access speed. In other words, each time the ALU carries out one operation, an average of one memory access is required. If the ALU's maximum processing speed is one operation per beat then a maximum memory speed of nearly one access per beat will be required. But because the core storage access cycle is T = 1,500 ns, meeting this requirement requires a modular interleaved parallel memory system. The main requirements for such a parallel memory system are an access rate of close to 1 access per beat (if the main frequency is 10 MHz, then one beat = 100 ns), and no conflict between units. Let us discuss these conditions.

a. For storage of an "equidistant" vector $\vec{A}$ (i.e., a vector whose increment Δ is a constant, if we designate the number of internal storage units in the parallel memory system as m, the access cycle for each memory unit by T (with the clock period P as a unit, 1P= 100 ns), and the address increment between vector components as Δ, we obtain the following results:

① When fetching or storing one vector the maximum number of storage units that can be accessed is:

$$m' = m/(\Delta, m)$$

where (Δ, m) is the greatest common factor of Δ and m.

② When accessing one vector in each storage cycle, if the vector length is $\ell > m/(\Delta, m)$, the maximum latency time is

$$t_A = \begin{cases} 0, & m/(\Delta,m) \geqslant T \\ T - m/(\Delta,m), & m/(\Delta,M) < T \end{cases} \qquad (1)$$

It is evident from equation (1) that for an "equidistant" vector the condition must be met for a collision-free parallel storage system is $m/(\Delta, m) \geqslant T$.

Because Δ is an arbitrary integer, this condition can be met only under certain conditions. Suppose Δ = 0, 1, 2, Km-1, (where K is an integer). If Δ is uniformly distributed on the interval from 0 to Km-1, then the average latency time is

$$t_A = \left(\sum_{\Delta=0}^{Km-1} t_\Delta \right) / Km$$

Because for all K = 0, 1, 2,

$$(\Delta, m) = (\Delta + Km, m)$$

therefore

$$t_A = \left(K \sum_{\Delta=0}^{m-1} t_\Delta \right) / Km = \sum_{\Delta=0}^{m-1} t_\Delta / m \tag{2}$$

Actually Δ is not uniformly distributed. For example, in multiplication of a matrix by a scalar,

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

If the elements of matrix A are stored row by row, and if vector $\vec{b}$ is a sequential vector, then for this operation $\Delta \equiv 1$. If we assume that the distribution probability for Δ = 1 is 75 percent, then equation (2) can be revised to

$$\begin{aligned} \bar{t}_B &= t_1 \times 75\% + \left(\sum_{\Delta=0}^{Km-1} t_\Delta \right) / Km \times 25\% \\ &= t_1 \times 75\% + \sum_{\Delta=0}^{m-1} t_\Delta / m \end{aligned} \tag{3}$$

The revised values can be read from line t_B in Table 1.

When accessing vector $\vec{B}$ after accessing vector $\vec{A}$, because the addresses are not continuous and the resultant unit collisions are random, the average latency time t_{AB} within each access cycle is

$$t_{AB} = \begin{cases} T(T-1)/2\,m & m \geqslant T \\ T - \dfrac{m+1}{2} & m < T \end{cases} \tag{4}$$

Because the 757 vector machine uses the lengthwise-crosswise processing method, the vector is segmented so that ℓ' = 16, which is equivalent to an additional latency time of t_{AB}/ℓ' during each access of a component. Because ℓ = 16, the individual time increments are considerable.

Table 1. Memory System Capabilities (Main frequency = 10 MHz, access cycle 1,500 ns = 15(P))

m / t,v	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
t_A(P)	14	13.5	12.67	12.25	10.8	11.5	8.86	9.63	8.22	8.7	4.91	8.58	2.92	5.79	5.2	4.81
V_A(MW/S)	0.67	1	1.56	1.83	2.8	2.33	4.1	3.58	4.52	4.2	6.72	4.28	8.05	6.14	6.56	6.79
t_B(P)	14	13.25	12.17	11.31	10.2	9.12	8.21	8.06	6.55	5.93	4.23	4.39	2.23	2.20	1.3	1.2
V_B(MW/S)	0.67	1.2	1.9	2.4	3.2	3.9	4.5	4.63	5.63	6.1	7.18	7.07	8.5	8.53	9.1	9.2
t_{AB} or t_{scalar}	14	13.05	13	12.5	12	11.5	11	10.5	10	9.5	9	8.5	8	7.5	7	6.56
V(MW/S)	0.7	1	1.34	1.94	2.70	3.44	5.06	4.8	5.22	5.65	6.80	6.70	8.18	8.22	8.84	8.92
m / t,v	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
t_A(P)	0.82	5.83	0.74	5.45	4.09	3.05	0.61	5.46	2.16	1.96	2.74	3.68	0.48	4.03	0.45	2.75
V_A(MW/S)	9.41	6.11	9.51	6.37	7.27	7.97	9.59	6.36	8.56	8.67	8.17	7.55	9.68	7.13	9.70	8.17
t_B(P)	0.21	1.46	0.18	1.36	1.02	0.76	0.15	1.37	0.54	0.49	0.68	0.92	0.12	1.01	0.11	0.70
V_B(MW/S)	9.86	9.02	9.88	9.1	9.32	9.5	9.9	9.09	9.64	9.67	9.6	9.4	9.92	9.32	9.93	9.6
t_{AB} or t_{scalar}	6.18	5.83	5.53	5.25	5	4.77	4.57	4.38	4.2	4.04	3.89	3.75	3.62	3.5	3.39	3.28
V(MW/S)	9.6	8.8	9.65	8.87	9.1	9.3	9.7	8.9	9.46	9.5	9.38	9.23	9.77	9.18	9.8	9.4

In the case of scalars or "nonequidistant" vectors ($\Delta \neq$ constants such as indirect address vectors), the latency time resulting from storage unit collisions is likewise random, and the average latency time is the same as in equation (4):

$$t_{scalar} = \begin{cases} T(T-1)/2\,m & m \geqslant T \\ T-(m+1)/2 & m < T \end{cases} \tag{6}$$

The average access speed of a parallel memory system for an "equidistant" vector is:

$$V_1 = V_{max} \cdot \frac{T-\bar{t}}{T} \tag{7}$$

V_{max} is the maximum speed of the system (i.e. the speed when there is no collision between units and there is one access per beat).

The average access speed for a scalar is

$$V_2 = 1/t_{scalar} \times 10^6 \text{ words per second} \tag{8}$$

If T = 1,500 ns, the principal frequency is 10 MHz, and 1P = 100 ns, then we have T = 15P. Substituting this value into equations (2), (3), (6), (7) and (8), and making the calculations for m = 1, 2, 3, ..., 32, we obtain Table 1 and Curve 1 of Figure 3.

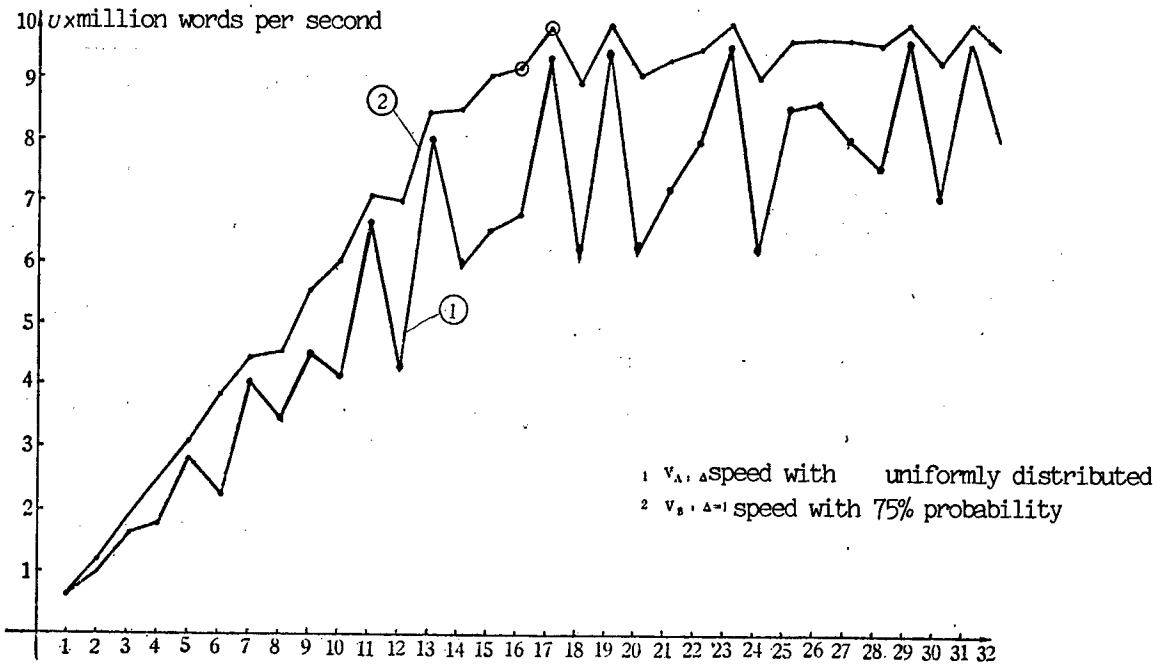


Figure 3. Memory System Capability, Main Frequency = 10 MHz, Access Cycle 1,500 ns

It is evident from equation (1) that for accessing an "equidistant" vector, in order to obtain a maximum speed of 1 byte per beat, we must meet the condition $m/(\Delta, m) \geqslant T$.

In the mainframe machine of the 757 vector machine T = 15P, so that the number of units m must be greater than 15. This is clear from Table 1 and Curve 1 of Figure 3. In addition, it is apparent that prime modulus numbers greater than 15, i.e., 17, 19, 23, 27, and 31 give a rather high-access speed. Of these prime moduli, modulo-17 is the most practicable; the other prime number moduli are unsuitable either because of complexity in address mapping or because of problems of memory use efficiency. But because of the factors noted below, the main memory system of the 757 vector machine uses a modulo-16 arrangement for the following reasons:

1. With modulo-17, it requires a rather complex address mapping structure;

2. Equation (4) indicates that when accessing vector $\vec{A}$ followed by vector $\vec{B}$, the additional latency time t_{AB} resulting from the fact that the addresses are not continuous is rather large in the 757 with a 16-segment lengthwise-crosswise processing. It is evident from Table 1 that when m = 16, T_{AB} = 6.56P. In other words, for every 16 segments accessed an additional 6.56P must be expended. If we use 16 segments, this time problem cannot be solved by using prime-number moduli. But if modulo-16 is used, then altered readout can be used to avoid loss of time T_{AB}. "Altered readout" means that when accessing vector $\vec{A}$ and then vector $\vec{B}$, if some memory unit is not locked, then the first unit that has been locked is accessed, and only after the 16

components have been read is the vector sequence restored in the look-ahead stations; this process increases access speed. The effect of T_{AB} is to decrease the memory system speed of a modulo-17 memory unit from 9.86 million words per second to 7.2 million words per second; since use of altered readout with a modulo-16 system gives a speed of 7.5 million words per second (assuming that the conditions for altered readout are met 40 percent of the time), it is obvious that the latter alternative gives a higher access speed than use of a modulo-17 memory system. Tests on the seven different problem types indicated that the altered readout method increases overall system speed by 13.7 percent.

3. It can be seen from equation (1) that if m = 16, when Δ is odd and $m/(\Delta, m) \geqslant T$ will always apply (when T = 15P), then it can be assured that there will be no conflict between units when accessing vector $\vec{A}$, so that a maximum speed V = 10 million words per second will be achieved. Under certain conditions the user can satisfy this condition (i.e., that Δ is odd).

The above discussion of hardware design has been sketchy and incomplete because many needed data have not yet been determined, compiled, and analyzed.

Participants in the development of the 757 vector machine's hardware included Lui Qiye [0491 0796 2814], Li Shuyi [2621 2885 1837], Yang Shufan [2799 2885 5400], Xia Shaose [1115 4801 3844], Xu Jun [1776 3182], Shi Guohua [4258 0948 5478], Li Changzi [2621 2490 1316], Luan Yumin [2940 3022 2404], Luo Yinfang [5012 6892 5364], and Xu Kunming [1776 2492 2494].

8480/9365
CSO: 4008/199

HARDWARE ENGINEERING OF 757 VECTOR COMPUTER SYSTEM

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 17-25

[Article by Luan Yumin [2940 3022 2404] of Institute of Computing Technology, Chinese Academy of Sciences (CAS)]

[Text] Abstract. A concise description of the 757's hardware engineering, implementation, and operation is given. This machine is the first to use Chinese-made medium scale integration [MSI] ECL [emitter-coupled logic] integrated circuits, large multilayer printed circuit boards and double connector printed circuit cards, low-impedance twisted pair lines, and long-line matching with three resistors, yielding complete 90-100 ohm impedance matching of the entire system, which minimizes ringing and crosstalk. Testing of small assemblies and more than 3 years' operation of the entire machine have shown that the hardware engineering system is reliable; the state evaluation indicated that the CPU was fault-free for 360 hours.

I. Introduction

The 757 machine is a pipelined vector machine with a speed of 10 million instructions per second. Users suggested many specifications for the machine, and in order to meet these requirements they were carefully taken into account in overall unit design, engineering design, and hardware design. Below we introduce the hardware engineering design.

The specifications for hardware engineering of the 757 which emerged from the overall design were a circuit delay of 4 ns, a logic chain of 17.5 levels, a maximum line length of 2 m, and a master clock frequency of 10 MHz. In order to meet these requirements we had to make the leads as short as possible, increase the degree of integration, and increase reliability. For this purpose, we developed three near-MSI circuits: a double D flip-flop, a quad half-adder, and an 8-bit shift register. In order to decrease lead length and achieve system matching, we used multilayer printed circuit boards and large cards, low-impedance twisted pair lines, single in-line package [sip] resistor networks, and high-frequency mica [?] [duli [3747 4539]]

capacitors. The machine contains 1,092 cards of 338 types, and a total of 57,178 integrated circuits (including backups). To date, the chip reliability is about 10^{-8}. In order to maintain reliable machine capabilities, we used 90-100 ohm impedance matching throughout the system, minimizing ringing and crosstalk.

In order to expose problems before production of the complete machine, we constructed a model unit with a word length of 16 bits, using 879 DIP [dual in-line package] integrated circuits of 10 varieties. These circuits were equivalent to 9,369 gate circuits and were mounted on 2 8-level printed circuit boards and 17 8-level large printed circuit cards, 1 control board, and 2 pairs of conversion plugs. After this model was tested for half a year and had shown no major problems, the complete machine was put into production. The process from board fabrication, card production, and PC board interconnection to the completion of automatic testing of the cards took nearly 2 years.

Because of a stringent effort in the key areas of production, the time required for debugging of the mainframe was greatly shortened; in particular, the process from testing of parts and components on printed circuit boards to the testing of fully assembled boards used automatic coded testing, so that all of the control units (instruction control unit, operation control unit, memory control unit, and ALU) were correctly debugged in less than half a year.

As a result of software debugging, the time required for the machine to objectively reach the testing stage was somewhat more than 2 years; the numbers of integrated circuits failing over the course of 2 years are shown in Figure 1. The two curves in the figure simply indicate how many modules failed in a given time interval. These failures all belong to the early failure type. Analysis indicated that most involved output follower or input terminal failure, in addition to which some were misdiagnosed; the circuit types in which problems were especially common were the shift registers and quad gates. The principal cause was insufficiently strict surface inspection. But shift registers are relatively heavily used in the instruction control unit, and in some places no heat sinks were provided, so that overheating resulted in a somewhat higher failure rate. Another 20 percent included mechanical damage and accidents.

II. Circuit Varieties and Characteristics

The circuits were divided into 10 categories: single gates, dual gates, voltage pulse gates, quad gates, power gates, receiver gates, double D flip-flops, 4-bit half adders, 8-bit shift registers, and reference sources. Two of these types (double gates and shift registers) are diagrammed below: the dual gates are shown in Figure 2 and the shift registers in Figure 3. The circuit characteristics are as follows:

① Cutting the emitter follower off from the emitter resistance can produce a "wired or" output, which expands the chip's logical capabilities and saves power, as well as allowing impedance matching with the system.

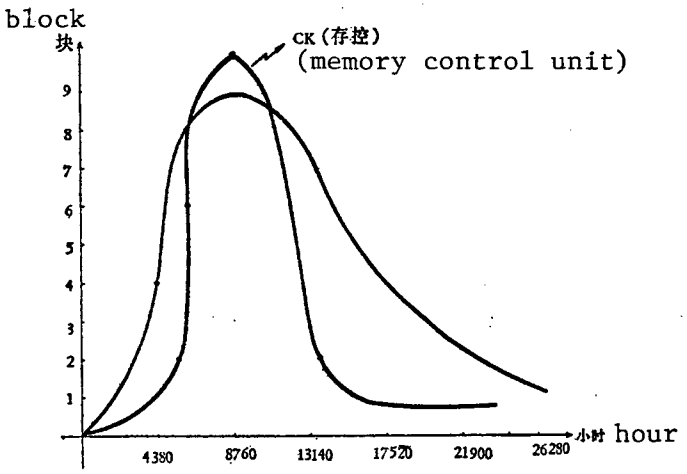


Figure 1. Time Frames of Failures in Memory Unit and Instruction Control Unit

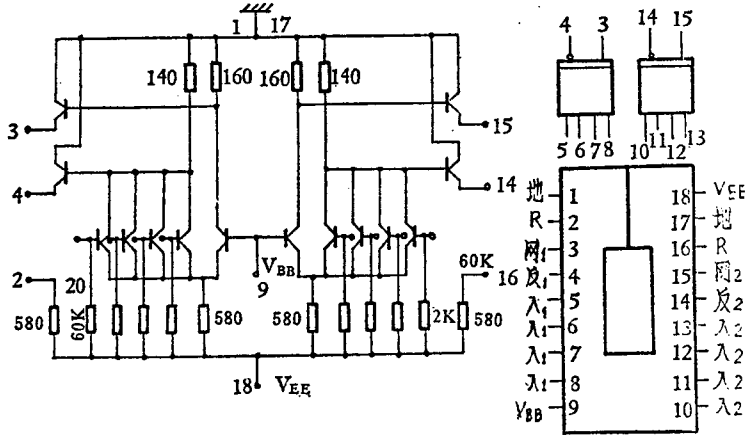


Figure 2. Dual Gate Circuitry

Note: Pins 1, 17 are ground; 3 and 15 are net 1 and 2; 4 and 14 are opposite 1 and 2; 5-8 are output 1; 10-13 are output 2

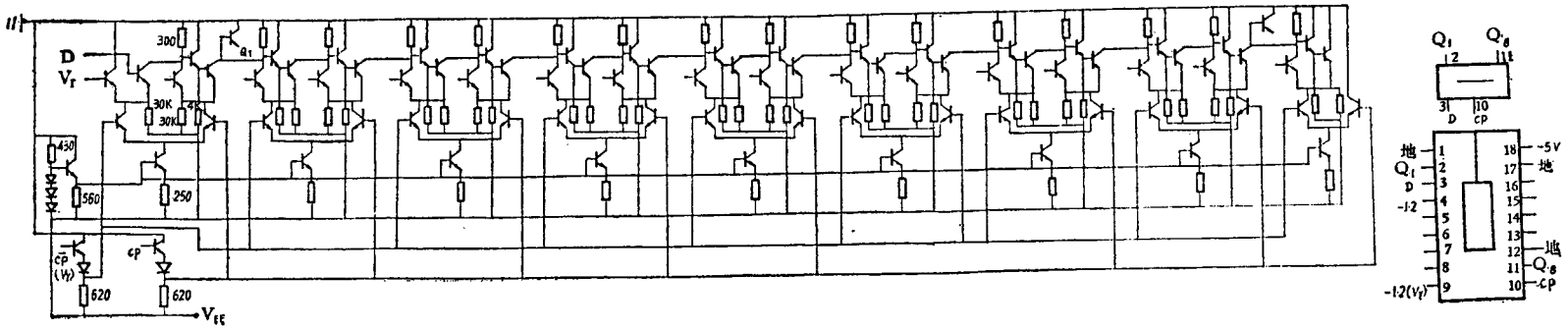


Figure 3. Circuitry of 8-Bit Shift Register

Pins 1, 12, and 17 are ground

② Each base input terminal has a base pull-down resistor, which avoids the low frequency effect resulting from input "float." This also decreases input capacitance and makes the changes on the output side in response to the load small.

③ Connection to an external reference source. Under current domestic production conditions, this helps to increase the chip acceptance rate and provides more possibility for flexible adjustment of the high- and low-voltage noise immunity. In addition it makes for convenience in measuring noise immunity, particularly for two-way inputs.

④ The three MSI circuits greatly increase the 757's gate-to-pin ratio and are key components in the 757.

⑤ Decreased power consumption, increased number of microscopic inspections, addition of a silicon nitride protective layer, rigorous screening, and testing.

III. Technical Characteristics

① The gate circuits and three circuit varieties described below operate at a voltage of -5 V. The power consumption is 25 mW per gate. The voltage level is from -870 mV to -1,750 mV.

② The double-D flip-flop uses a two-layer current switch. It has 62 transistors and 32 resistors. In terms of capability this is equivalent to 12 gate circuits. The maximum operating frequency is 200 MHz, the average delay is less than 8 ns, and the power consumption is 100 mW.

③ The 4-bit half-adder is the key component in the 757 machine's addition tree and checking circuitry. It too uses a double-layer current switch, and each half-adder has 50 transistors and 18 resistors, with an average delay equal to or less than 5 ns.

④ The 8-bit shift register shown in Figure 3, can replace random-access memory. Each register consists of 87 transistors and 62 resistors and has an operating frequency of 100 MHz. It has a capability equivalent to 96 gate circuits, its power consumption is less than 200 mW, and the average delay is less than 8 ns.

⑤ Circuit noise immunity. As is generally known, ECL circuits can use low-resistance transmission and matching. In the transmission process we specified that the noise immunity at both high and low levels should be 150 mV. The results of more than 2 years of machine operation indicate that these requirements were essentially met.

⑥ The devices are categorized according to voltage levels and are selected for use strictly in accordance with this classification. The changeover region of the gates is generally 250-300 mV, and the reference voltage tolerance is ±25 mV. The central value of the reference voltage is specified as -1,200 mV and there is normal switching guaranteed to be between

-1,050 and -1,350 mV. Power supplies of -5 V and -2 V opposing voltage [?] [duila] [1417 2139]] by 10 percent of the measurements. In actual use, the packages are classified and used with reference to load conditions.

(7) Transmission. Transmission matters are a very important topic in high-speed digital systems. When a signal is transmitted along a line, it does not merely consume energy (because transmission lines have interface losses) and increase delays; in addition, in an incompletely matched system it also encounters multiple reflection and crosstalk, which affect the system's stability and reliability. The transmission paths in the 757's circuits run from the card interconnections and connectors to the printed circuit boards; those more than 10 cm long generally use 95-ohm twisted pair lines before being connected to the racks via connections and long lines. In order to minimize reflect and crosstalk, 95-ohm impedance matching is used throughout the system. The actual measured resistances of the printed circuit boards are 80-90 ohms, and the high temperature resistance of the paired wires is 95 ohms, while the chip pulldown resistances are 95 ohms. Thus the entire system other than the connectors has an essentially continuous, matched impedance, so that the reflection coefficient is very small. The measured waveform in the machine (Figure 4) shows that the system basically has complete matching, and wave distortion is very small.

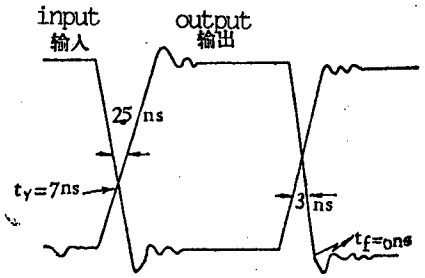


Figure 4. Actual Waveform for Machine With Seven Loads

IV. Printed Circuit Cards and Boards

In order to contain more information lines, to increase layout density, to decrease the length of interconnections and to make transmission easier and help assure uniform system impedance, we used printed circuit cards and boards.

1. Requirements for Printed Circuit Cards

(1) The electrical characteristics of printed circuit cards must meet signal continuity requirements, have uniform, consistent impedance, be matched to the system impedance, and must have good grounding and power supply systems in order to suppress noise and to keep all card noise within the permitted range.

(2) The layout density and gate-to-pin ratio must be maximized and card metallization requirements must be minimized.

(3) Account must be taken of overall machine layout and its requirements.

(4) Design should be based on card fabrication technology and every effort made to increase product acceptance rates and assure quality and reliability.

(5) Provision should be made for automating logic partitioning and the production process, and suitable provisions should be made for debugging.

2. Characteristics of the 757's Printed Circuit Cards

(1) Dimensions and Structure:

Measurements are 175 x 278 mm². A card has two 72-line printed connectors (a total of 144 lines), of which 36 are grounded and 108 are used for signal input and output. There are 50 inspection holes in the upper left corner, and the card has 5 rows of 18-pin DIP packages, each row divided into 10 or 16 chip areas, including 2 voltage reference chips. Below each chip is an 0.01 μF high-frequency filter capacitor to filter the reference voltage, and a matching resistor network is mounted between each two lines of chips and outside the rightmost and leftmost chips; these are six-line single in-line [SIP] packages. Below the resistor network is installed an 0.01 μF high-frequency filter capacitor. The -5 V and -2 V power supplies have 20 to 65 high-frequency filter capacitors, respectively, and a low-frequency high-capacitance 30 μF clamping capacitor is installed at the -5 V and the -2 V input: thus every board can have a maximum of 80 18-pin DIP chips and 85 6-pin SIP resistor networks, 165 0.01 μF high-frequency filter capacitors and 2 low-frequency capacitors. The card layout is shown in Figure 7.

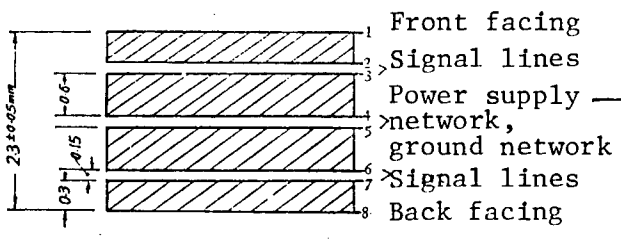


Figure 5. Layered Printed Circuit Board Structure

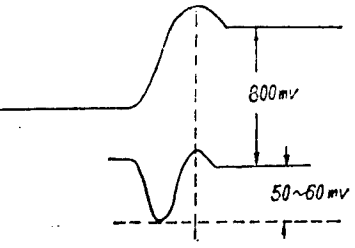


Figure 6. Noise Waveform

Note: The lower voltage level in the 800-mV interval is the lower limit of the waveform shown

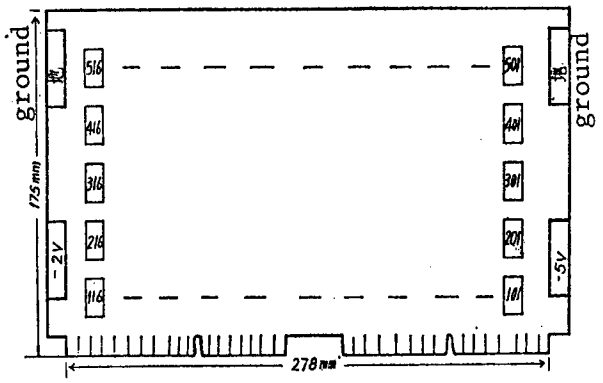


Figure 7. Card Layout

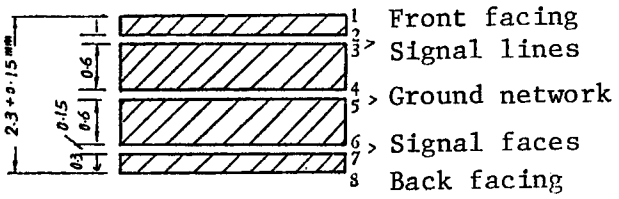


Figure 8. Layered Structure

(2) Layered Structure

If the printed circuit resistances are uniform and if the ground and power supply networks have sufficient capacity, a rather large number of working paths and a rather high layout density can be achieved. The 757 machine's printed circuit boards have eight layers, with the top-most and bottom-most layers as panels. On one side of the components is a -2 V power supply network, while only solder pads are on the other side; signal connections are run on layers 2, 3, 6, and 7, while the ground and power supply networks are on layers 4 and 5, with the -2 V and -5 V power supplies on the same layer. The boards are 2.3 ± 0.15 mm thick and the metallized holes have a diameter of 0.9 mm. The overlap structure is shown in Figure 5.

(3) Hole Layout and Interconnection Method:

The chips used in engineering the 757 were 18-pin DIP components with a pin spacing of two lines of pins 7.5 mm apart and a pin spacing of 2.5 mm. The hole spacing used on the cards is 2.5 mm x 3.75 mm, with the 72 connectors spaced 3 mm apart. Problems involving conversion to the internal network are solved by card layout. The internal vertical and horizontal lines are 1.25 mm from the holes. The holes with a spacing of 2.5 mm each have one cross line, and those with a 3.75 mm spacing each have two lines. The lines are 0.2 mm wide.

(4) Electrical Parameters:

Experimentally determined parameters: C = 80-90 pF/m (capacitance per unit length). L = 0.6 - 0.8 μH/m (inductance per unit length). Z_0 = 80-90 ohms (impedance of printed interconnections; T_d = 0.075 - 0.8 ns/cm (no-load delay); Rd = 0.02 ohms/cm (DC resistance of signal lines).

The interconnections' characteristic impedance of 80-90 ohms is in relatively good agreement with the system impedance, but in the presence of capacitance it is somewhat decreased. The equation is $Z_0' = Z_0 \frac{C_0}{C_0+C_d}$ (empirical formula),

where C_0 is the intrinsic capacitance, C_d is the capacitance under load, and Z_0' is the impedance in the presence of capacitance. The no-load delay T_d is 0.075 - 0.08 ns/cm; the delay is higher when a load is present: $T_d' = T_d \cdot \sqrt{1 + \frac{C_d}{C_0}}$. Currently, the load value is $T_d' \leqslant 0.1$ ns/cm.

(5) Noise Problems of Printed Cards

① Ground Connection Noise: If there are 60 simultaneous transitions on the card, the maximum noise on the ground connections is 50-60 mV. The noise waveform is shown in Figure 6.

② Contact Pin Noise: In order to combat contact pin noise, it was decided to ground one-fourth of the contact pins in the 757 mainframe, so that the ratio of ground pins to signal pins is 1 to 4. When a certain signal pin is in a static state and the six signal pins around it transmit signals of the same change simultaneously, the noise induced in the static signal pin does not exceed 90 mV. The actual circumstances are better than the experimental circumstances (see Figure 12).

③ Crosstalk Between Parallel Wires: When two parallel transmission lines are close together, the capacitance between them and their mutual inductance may produce crosstalk. The measured interference between overlapping parallel lines in adjoining layers is the most severe type of interference between lines, and the maximum crosstalk figures are as follows:

Length of overlap and parallelism	Maximum crosstalk voltage
10.75 cm	270 mV
12.75 cm	290 mV

On average, for each centimeter of parallel overlap, the crosstalk can be as great as 25 mV. Accordingly, overlap must be minimized.

3. Printed Backboards

(1) Dimensions: 234 x 300 mm^2, contain 24 72-line spring-loaded receptacles and can hold 12 cards. There are 91 connector holes on the right and left.

(2) Layered structure: Same as for cards, except power supply network is replaced by ground network (see Figure 8).

(3) Wiring paths:

a. With a hole spacing of 3 mm, one line can pass between them;
b. with a hole spacing of 4.5 mm, two lines can pass between them;
c. with a hole spacing of 3.75 mm, two lines can pass between them. Other electrical characteristics are the same as for the cards.

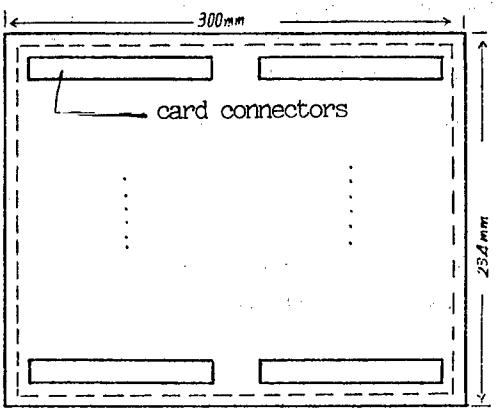


Figure 9. Printed Backboard Layout

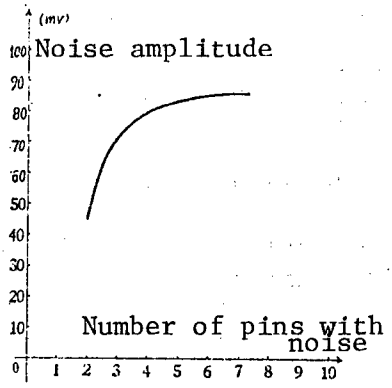


Figure 10. Card Noise

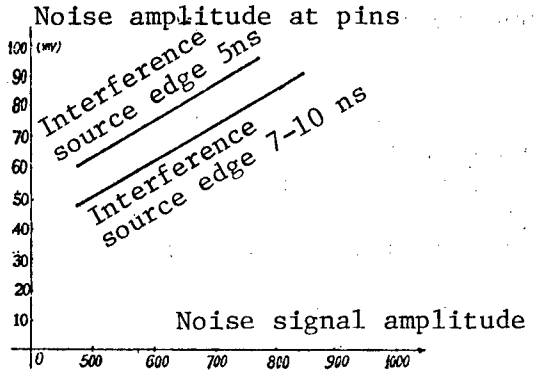


Figure 11. Amplitude Change of Noise Signal for Different Noise Source Amplitudes and Edge Changes

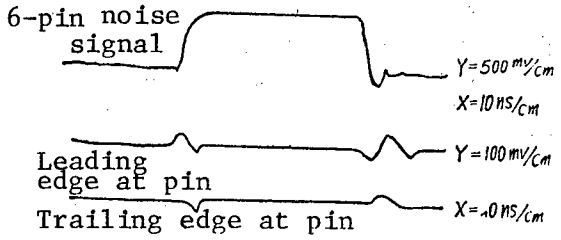


Figure 12. Noise Signal at Pins

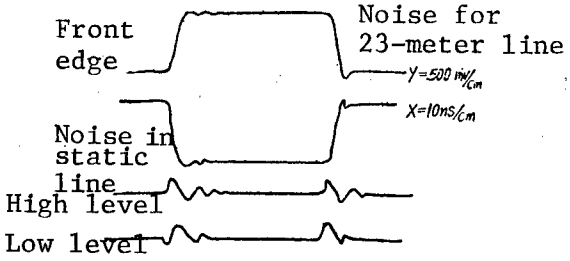


Figure 13. Noise Waveform for Long Line

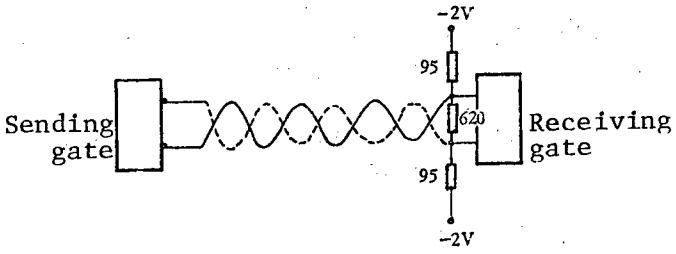


Figure 14. Use of Three Resistors To Replace Two-Resistor Matching

Each layer has 90 paths for transverse lines and 96 paths for vertical lines, and each card can hold a maximum of 1,000-plus lines (actual statistics). The surface layout of the card is shown in Figure 9.

4. Tests of Card Contact Pin Noise

Tests of card noise indicate that for six connectors or more the amplitude of the noise signal is constant (see Figure 10).

The relationship between the interference edge and the amplitude of the interference signal received at the pins can also be seen. As Figure 11 shows, the quicker the edge, the greater the interference signal.

V. Tests With Long Lines

In computers and control systems using large numbers of digital circuits, the connections within an individual circuit must follow certain logical requirements. Some of the interconnections are rather short, such as those between neighboring chips on a card, but some are rather long, such as connections between frames. Tests show that the transmission speed in some electric circuits is very high, close to the speed of light. The shortness of this transmission time clearly is not critical for low- and medium-speed digital devices, but as highly integrated circuits have been developed and put into use, the situation has changed. For example, the gate delay of high-speed ECL circuits is 2-4 ns, or even as little as 1 ns. In this case the transmission delay is much greater than the gate delay. But the problem is not simply the transmission delay: a more important factor is that when a rapidly changing signal is transmitted along a long line, reflection may occur, seriously distorting the signal waveform and producing various harmful interference pulses which hinder the normal operation of the entire system. Therefore, in high-speed digital computers, signal transmission has become an important problem that must be solved during design. We call the interconnections in which transmission delay must be taken into account "transmission lines" or "long lines." In engineering the 757, transmission lines 2 meters long or more were regarded as long lines. The wave resistance in the 757's long lines has been found to be 150 ohms, and the transmission delay is 5-6 ns per meter. Here we focus on solving the following problems: 1) loss problems: if we use a 5-meter line with load, the waveform obtained when a signal with an edge of less than 8 ns is used to drive it, the measured waveform agrees with that derived from the Bessel function, and accordingly in analyzing long lines this can be regarded as a lossless line. 2) Basic

$$L_0=\frac{L}{M}=(7.3-7.5)\times 10^{-1}(\mathrm{MH});\ C_0=\frac{C}{M}=(45-49)\mathrm{pf};\ Z=130\ \Omega\sim 140\ \Omega$$

parameters: inductance and capacitance per meter. 3) Signal transfer method: bidirectional transfer, i.e., conjugate form. 4) Matching method: here we discuss matching in which three resistors replace two, as shown in Figure 14. We calculate R_1 = 95 ohms, R_2 = 620 ohms. In order to save space, we construct a matching network. This type of matching produces good results with little waveform distortion and little loss of level. The reflection coefficient is extremely small.

We also made crosstalk tests which demonstrated that with a group of 10 lines, the noise waveform on one of them which is quiet while the other 9 are carrying signals is very small, no greater than 50 mV (see Figure 13).

Twisted pair lines with biphase transfer were designed for an impedance of 150 ohms. The bandwidth of the lines is 80 MHz, and there is essentially no amplitude loss in them because the resistance of multi-leg lines is small.

VI. Design of Twisted Pair Lines

Twisted pair lines are one type of transmission line used in recent high-speed computers and communications networks. Although they are inferior to coaxial cables in frequency terms, their wave resistance is high, they are of small size and flexible. In addition, twisted pair lines are particularly suited to complementary signal transmission, and the crosstalk between them is extremely small. But the problem of how to design such lines to give the impedance requested by the user, i.e., how to choose their external and internal diameters, is still a difficult one. However, we finally achieved a solution through analysis and computation.

VII. Transmission Requirements

In order to make the most rational use of chips, the line noise is kept within the permissible range; the noise immunity is considerable. We analyzed the multiple reflection of ECL signals in transmission lines and derived the formula relating the voltage at the terminals and the load to time and various parameters when the lines are driven by ECL circuits and they are affected by both concentrated and distributed loads; we used a computer to predict the waveform, achieving excellent results.

Some of our results were as follows: Because current switches can provide "wired OR's," we specify that all output terminals forming part of wired OR's and their loads must be distributed on a transmission line in order to prevent reflection, and cannot branch. In general, the maximum permissible distance between "dotted OR's" is 80 mm. In all circuits we specify that the load factor must not exceed eight. Matching resistors must be provided at terminals. In general, only one driver card is allowed; the maximum is two. The total number of loads also must not exceed eight, and so on. We will not list more detailed prescriptions here. Because a determined effort was made in this area during implementation of the machine, the overall operation of the machine is excellent.

VIII. Operating Principles of ECL Circuits

The ECL gate circuit is a differential amplifier type and its current is essentially constant. If one terminal is connected to a reference source and the other terminal to a dynamic signal, the signal will fluctuate by 400 mV above and below the reference voltage. In high-level output, the input channel is closed. When the input is low, the reference channel is closed. The output is driven by a follower, and it can supply a low internal resistance, high drive capability load. Measurements on such gate circuits under dynamic conditions indicate that the gate delay at each level is 2 ns and the power consumption 25 mW, and with a power supply of -5 V. The follower resistance is connected externally. The 757 used a 95-ohm matching network. The forward edge and rear edge are both 6-8 ns (with

eight loads). The Boolean experession is: "$\bar{3}$" = $\bar{5}\cdot\bar{6}\cdot\bar{7}\cdot\bar{8}$ (for the case of a double gate); see Figure 2.

The ECL flip-flop is a double D flip-flop; two flip-flops are included in each package and there is a pulse input terminal, a reset "0" terminal and two set "1" terminals. Every flip-flop has two data input terminals. The usual Boolean expression is $Q^{n+1} = C_p(D_1 + D_2 + \bar{R}\bar{S}) + \bar{C}_p Q^n$, where C_p is the pulse input, R is the reset terminal, and S is the set terminal. D_1 and D_2 are the two data input terminals. Q^n is the device state set by the previous pulse, and Q^{n+1} is the device state during the present pulse.

A half-adder is a device which is capable of summing and carrying. Like the flip-flop, it is also a tower [?] [baota [1405 1044]] current switch, whose lowest level is DC, and the logical operation which it carries out is $A\bar{B} + \bar{A}B$, where S is the half-sum, A and B are the input signals, and C = AB, where C is the carry bit. In the present case, because of a limited number of connections, C did not output. Every package in it has four half-adders. The delay is 4-5 ns at each level.

A shift register contains 8 bits, all of which are in a single package, and only the first and eighth bits (called Q_1 and Q_8) are output. Each bit has an input voltage, but there is no set or reset signal, and it is only possible to enter a 0 or 1. Clearing an 8-bit shift register requires nine counts. To change all 8 bits from 0 to 1 requires a 1 input for 9 counts. There is an 8-ns delay for each bit in the shift register; the Boolean expression for Q_1 is $Q_1^{n+1} = C_p D_1 + \bar{C}_p Q^n$ (see Figure 3).

IX. Conclusions

To summarize our experience, the 757 machine provides a 17.5 level logic chain and 2 meters of lines with a 100-ns cycle, and there is only a 10-ns margin after computation, which is somewhat too small. In addition, the 2 meters of line is hard to achieve at current domestic levels of integration.

The foreign MU-5 is also an ECL circuit, with a cycle of 50 ns, whose logic chain only permits 8 levels, each with a delay not exceeding 4 ns, with the lines included within the eight levels. The ILLIAC-IV has a cycle of 67 ns and the lines and delay are specified as 5-15 levels, with 20 levels permitted under extraordinary conditions. The margin after computation is 30 percent in the MU-5; that in the ILLIAC-IV is 25 percent.

The 757 machine has a margin of only 10 percent, so that currently it is operated normally at 110 ns. In order for machine operation to be reliable, 8.5 MHz is specified. It should be pointed out that most of the gate delays in chips involved in transmission are 2.5-3.5 ns, and the delay per meter of printed circuit connection is 8 ns, while that per meter of twisted pair line is 6 ns. These values are all smaller than the figures given in the engineering standards. If the total delay of a chain measured in the ALU is 110 ns and the circuit delay is only 49 ns, the line length exceeds 5 m. This

indicates how harmful the delay in the wiring is, since it is hard to keep it within 2 m.

Furthermore, the project specifications state that in order to make the boards capable of accommodating more wiring, it is desirable that their size be further increased, (some difficulties are currently being encountered in the process). In the case of cards, we distinguish type A boards (no double rows of holes for device pins) and D type (double rows of holes for device pins): in changing the wiring, it is desirable to use more D-type cards, but this decreases packaging density. The problem of how to design future printed circuit board substrates and boards requires examinations.

Research and development on the 757 machine promoted the development of semiconductor and electronic component manufacturing processes. Because the 757 machine largely makes use of automation techniques, CAD [computer-aided design] was developed.

Those involved in the 757 machine hardware engineering project also included Comrades Zhang Shuwen [1728 3219 2429], Wang Gonghao [3769 0361 8504], Liang Peiji [2733 1014 1015], Chen Hong'an [7115 7703 1344], Huang Zhenyun [7806 6297 5619], Xu Xingsheng [1776 5281 5116], Liu Pixuan [0491 0012 4821], Liao Qingyu [1675 3237 5940], Li Wenyin [2621 2429 6892], Sun Xiangjun [1327 4382 7486], Lu Pinghe [4151 1627 0735], and Jin Lianyi [7246 6647 5030].

8480/9365
CSO: 4008/199

DESIGN, CHARACTERISTICS OF 757 VECTOR COMPUTER MEMORY CONTROL UNIT

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984, pp 26-33, 16

[Article by Xia Shaose [1115 4801 3844], Institute of Computing Technology, Chinese Academy of Sciences (CAS)]

[Text] The 757 machine system uses an architecture consisting of a mainframe and an auxiliary machine, between which information is exchanged by two methods. In the first, the communications channels are used to transmit control signals in the interrupt mode, while the other uses a fast batch transmission channel in which data is transmitted in batch form in the time-sharing mode. The 757 vector machine's memory control unit controls the communications channels between the mainframe and the auxiliary machine and exchange of signals on the fast batch transmission line. In addition, the memory control unit is connected by instruction or data channels to the instruction control unit and the operation control unit. In short, the 757 machine's memory control unit carries out information exchange between the vector processor, the peripheral processor, magnetic disk (or tape) and the vector processor's main storage. Seven channels can make simultaneous requests to the memory control unit, which performs time-share queuing, address processing, and address mapping in accordance with priority, processes instructions in accordance with functional requirements, and accesses the relevant memory until processing of the requests is completed.

The 757 vector machine uses a parallel interleaved single-pipeline architecture, and the memory control unit is geared to this type of operation. In order to further increase the machine's speed and efficiency, to make full use of the pipeline architecture, and to minimize speed degradation resulting from collision problems, the memory control unit has been provided with a collision processing capability, and the altered readout technique. In order to increase machine reliability, in addition to fault diagnosis of the memory control unit, instruction-level retry and memory-level double calculation are provided. In addition, the memory operates in modular fashion and performs such measures as switchovers, disconnection of units, address mapping, and the like. Because of the use of these technologies and features, the memory unit economizes on hardware, and has a rather high speed and relatively high efficiency.

Below we briefly describe some of the principal problems involved in memory control unit design and the unit's design characteristics. For convenience in dealing with the topic, we first outline the memory control unit's architecture and mode of operation.

I. Structure and Mode of Operation of the Memory Control Unit

1. Time-Sharing Queuing of Requests

The memory control unit operates in the pipeline mode with a seven-station pipeline. Time-sharing queuing of requests is done at stations 0 at the beginning of the pipeline. The memory access requests from different locations are queued according to priority and are handled in the time-sharing mode in accordance with the memory control unit's clock pulses. This time-sharing processing of requests by beat means that although requests from the various channels may appear randomly, only one request can be processed per beat; the top-priority request is forwarded to the following stations of the pipeline. In order to increase the rate of information exchange between the peripheral processor (the auxiliary machine) and the vector processor (the mainframe), direct fast batch data transmission channels are used. Channels P_0 and P_1 (disk 0 and disk 1) are connected to a 2 MHz magnetic disk unit, and because the exchange signal is timed, P_0 and P_1 are given the highest priority. In addition, there is an instruction fetch channel, a data send store channel, a fetch and send channel, and two peripheral processor channels, as shown in Figures 1 and 2.

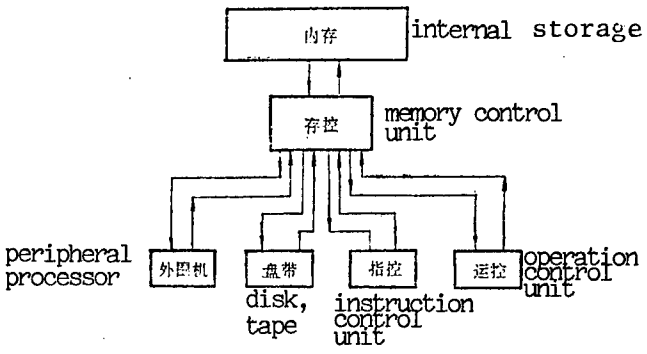


Figure 1. Memory Control Unit External Connections

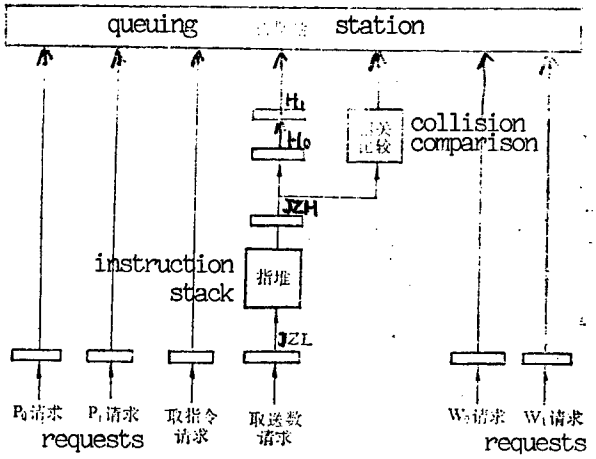


Figure 2. Request Queuing Station

To adapt the device to magnetic disks and increase the main frequency, the memory control unit has two disk buffer registers and read-write control circuitry.

2. Fetch-Send and Send Channels

These two channels include an instruction fetch and send stack station, two data send instruction holding registers (H_0 and H_1), and a collision comparator. The data send instruction read out from the instruction stack requires look-behind data which has not yet been processed and prepared in the ALU, the data send instruction is placed in H_0 and H_1, and the next data instruction is compared with these two data-send instructions which are in the wait

condition; if there is no collision, it can be moved ahead in the queue for execution.

3. Pipeline Stations

Requests from all of the channels require exchange of an array. Generally, only the initial address forwarded after instruction processing, the address mapping method, the increment, and the number of exchanges are processed by the memory control unit. In accordance with the relevant address mapping method, the memory control unit finds the logical address of each component, then looks at the page table to find the real address; then, depending on whether the memory unit is in the switched-in or disconnected state, it converts it to the true internal storage access address.

In accordance with the capabilities of the 757 vector machine, in order to enrich the vector processing methods and expand the range of vector computations, the compression and restoration vector $\vec{h}$ and the indirect control vector $\vec{Q}$ have been provided, and there is a requirement for two different address-forming methods: ① scalar, $D' = D$ or $D' = D + (Q_iB)$; ② vector, $\vec{D}' = D + 1'(\vec{BE})$ or $\vec{D}' = D + (\vec{Q}_i)$.

Thus the component increment (BZ) or the indirect control vector (Q) can be used to modify the address. In addition, it may be controlled by the compression and restoration vector ($\vec{h}$). When the address modification makes use of the component increment (BZ), $\vec{h}$ serves to control compression and restoration; but if the instruction is controlled by both the direct address ($\vec{Q}$) and by $\vec{h}$, then $\vec{h}$ determines whether or not the fetch or storage operation is carried out.

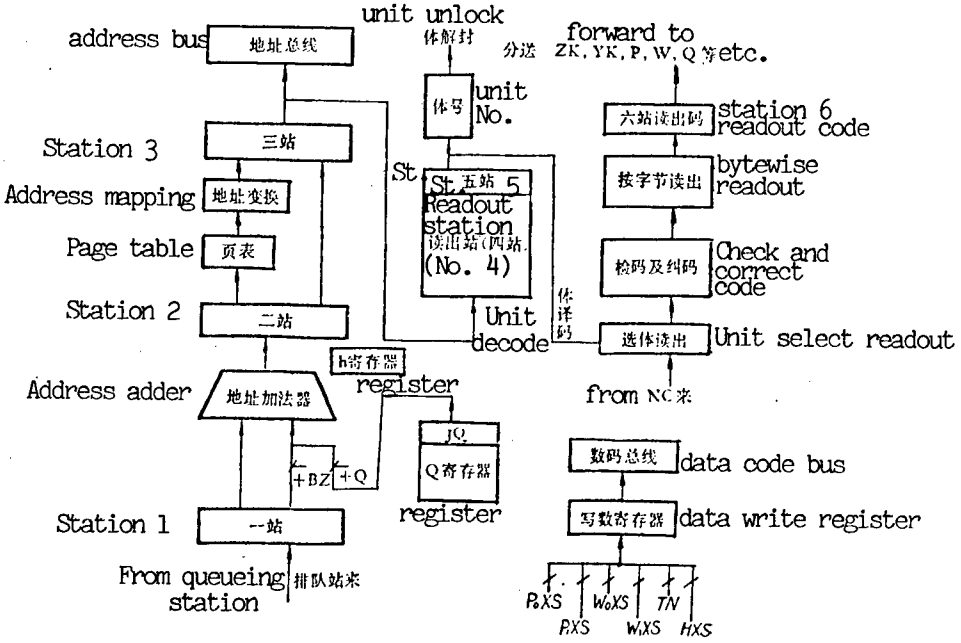


Figure 3. Rough Block Diagram of Memory Control Stations 1-6

An example:

	$\vec{h}$	$(\vec{L}) \xrightarrow[\text{compress}]{\vec{h}\ \text{control}} D+(\vec{BZ})$	Result	$\vec{h}$	$(D+(\vec{BZ})) \xrightarrow[\text{restore}]{\vec{h}\ \text{control}} \vec{L}$	Result $(\vec{L})$
$\vec{D'}=D+(\vec{BZ})$	0	x_0		0		0
	0	x_1		0		0
	1	$x_2 \longrightarrow D$	$\longrightarrow (D)=x_2$	1	$(D)=x_2 \longrightarrow L_2$	x_2
	0	x_3		0		0
	⋮	⋮		⋮		⋮
	1	$x_8 \longrightarrow D+(BZ)$	$(D+(BZ))=x_8$	1	$(D+(BZ))=x_8 \longrightarrow L_8$	x_8
	0	x_9		0		0
	⋮	⋮		⋮		⋮
	1	$x_{14} \longrightarrow D+2(BZ)$	$(D+2(BZ))=x_{14}$	1	$(D+2(BZ))=x_{14} \longrightarrow L_{14}$	x_{14}
	0	x_{15}		0		0

	$\vec{h}$	$(\vec{L}) \xrightarrow{\vec{h}\ \text{control}} D+(\vec{Q})$	Result	$\vec{h}$	$(D+(\vec{Q})) \xrightarrow{\vec{h}\ \text{control}} L$	Result $(\vec{L})$
$\vec{D'}=D+(\vec{Q})$	0			0		0
	0			0		0
	1	$x_1 \longrightarrow D+(Q_1)$	$D+(Q_1)=x_1$	1	$(D+(Q_1))=x_1 \longrightarrow L_1$	x_1
	0	x_2		0		0
	⋮	⋮		⋮		⋮
	1	$x_8 \longrightarrow D+(Q_8)$	$D+(Q_8)=x_8$	1	$(D+(Q_8))=x_8 \longrightarrow L_8$	x_8
	0	x_9		0		0
	⋮	⋮		⋮		⋮
	1	$x_{14} \longrightarrow D+(Q_{14})$	$D+(Q_{14})=x_{14}$	1	$(D+(Q_{14}))=x_{14} \longrightarrow L_{14}$	x_{14}
	0	x_{15}		0		0

It can be seen from Figure 3 that stations 1 and 2 are provided to complete the above two types of address modification. From station 2 to station 3, the page table is consulted and the address mapping is made in accordance with the switched-in or disconnected state. To adapt to the machine's multiprogramming operation and memory protection features, the memory control unit has a page table memory with a capacity of 128K: each page has 4,096 internal storage cells. The word length in the page table is 14 bits, of which 7 give the actual page number, 5 give the protection key, and each of these divisions has a parity bit. The machine has three modes of operation, i.e., kernel, supervisor, and computation, and there are three protection states: read enable, write enable, and execute enable. The page is provided with protection rules, and if these are violated a page fault interrupt is generated (YBBF).

From stations 3 to 4 in the pipeline, the internal storage unit number and true address resulting from address mapping are sent simultaneously via the address bus to the 18 units, while the access signal is sent to only one of the units, and that unit is locked. If it is decided to write into main memory, the data code is simultaneously sent to the 18 units by the data write buffer. This type of bus operation saves considerably on hardware. Station 4 is used to record instructions, byte characteristics, the number of the internal storage unit access, the characteristics of the read-write commands and the like. Eight shift registers provide a delay while waiting for the internal storage readout.

From station 5 to station 6, a number read out from memory is received, undergoes byte conversion and is sent to various channels. In addition, this datum read out of internal storage is diagnosed and the code corrected. A single bit error can be corrected, after which the correct code is sent out bytewise. In station 2, a check is made to see whether the addresses of the two immediately adjoining fetches are the same; if the data is fetched in bytewise fashion and the preceding and following component fetch cells have the same address, another read from memory is not required: the unlocking of the unit is simply delayed until the two addresses are different. This saves time and increases efficiency.

II. Collision Processing

The 757 mainframe is a single-instruction flow, single-data-flow lengthwise-crosswise vector processing machine. The control components all have pipelined architecture and all operate with parallelism and overlapping. The main obstacle affecting the smooth parallel overlapped operation of the three control units' pipelines is the various collision problems such as jump collisions, operation collisions, data fetch and send collisions, storage unit collisions and the like. Here we limit ourselves to describing a few of the collision problems encountered in the 757 vector machine's memory control unit and means of resolving them. Memory control unit collisions primarily entail collisions within the instruction flow or data flow and relations between the instruction flow and data flow. These factors affect normal use of program states, memory unit states, page contents, and the fetching and forwarding of internal storage addresses. Handling of collisions must above all be logically correct, and in addition must maximize the machine's speed and efficiency. Some problems which are difficult to handle in hardware can be handled by software.

1. Collisions Between Channels

The channels operate in the time-sharing mode, and the disk and tape channels exchange arrays with internal storage in batch form via the memory control unit, generally using true addresses (without protection); the peripheral processor has the right of initiation. The instruction fetch channel's execution program is edited and processed by the peripheral processor. Therefore, when internal storage addresses are used in distributed fashion, the user has to provide for this in advance, and the software processes the collisions.

2. Collision With Machine Status Word

When a transfer to supervisor instruction is encountered in the instruction flow (exchanged between internal storage and central numbering register), processing before and after this instruction is in accordance with different program status words and memory unit states. Before this transfer to supervisor instruction is executed, the machine stops dispatching the next instruction and executing the next program, and the instructions remaining in the instruction buffer are discarded. After this transfer to supervisor instruction is executed, a new instruction is fetched in accordance with the new machine status word and executed.

When a page table send instruction is encountered in the program, following this instruction, internal storage is addressed in accordance with the new page table. In general, changing of the page table state is controlled by the operating system, operating in the kernel state, and after the page table has been sent, a change of state instruction must follow; operation then continues in accordance with the new page table state.

3. Data Fetch and Send Collisions

The 757 machine is a crossbar vector machine, and a vector instruction may have from 1 to 16 components. The maximum number is 16, and the data fetch and send address is obtained by first having the instruction controller determine the real address by the operation d + (b), after which it sends this starting address to the memory control unit, which then must use various modification methods, i.e., use of the component displacement (BZ) or indirect control vector (Q) to process this address. In order to assure correctness and increase efficiency, collision processing is peripheral after the instruction reaches the memory control unit and before request queuing. But at this time there is no way of knowing the specific address of the 16 components following modification, and the specific content of the indirect control vector cannot be read out, so that decisions on address collisions are rather difficult and complex. Because the operand read out of internal storage is sent to the ALU via the memory control unit and the result obtained by the ALU is sent to the instruction control unit's look-behind data send station, then returned to main storage via the memory control unit, the entire flow path is long, and the data send results are not obtained very rapidly. Thus, if the data fetch and send is carried out in sequence, this seriously degrades the speed of the mainframe. Therefore, collision processing must be done for data fetch and send, so that fetches and send operations which involve no memory space collision can be carried out first; although this method requires a slight increase in hardware, machine simulations indicate that collision processing on data fetch and send by the memory control unit can increase machine efficiency by 30 percent or more. Moreover, measurements on the seven different problem types indicate that collision processing decreases computation time and yields a relative increase of 65 percent in speed.

Instructions are generally executed in sequence, but if a look-behind data send instruction is encountered, and the datum is not yet ready at the look-

behind station, the data send instruction is placed in data send wait stations H_0 and H_1. If the subsequently arriving data fetch and send instruction has no collision with the previous data fetch and send instruction, it can be moved up in the queue and executed. Instructions in which there is a collision should be placed in the instruction stack; and the later instruction in which there is a collision are executed after the earlier data send instructions are completed. The following four situations can arise with data fetch and send instructions:

	Earlier instruction	Later instruction
①	send	send
②	send	fetch
③	fetch	send
④	fetch	fetch

In situation ①, the look-behind instruction fetch obtained from the computation result is processed in strict sequence. If a data send instruction which does not use the look-behind buffer appears, the collision comparison is made in terms of the internal storage addresses used, and if there is no collision, it may be moved up and executed. In situation ②, if the following fetch instruction involves no collision with the internal storage address of the preceding data send instruction and the Q (indirect control vector) or h (compression and restoration vector) is used, it can be executed ahead of sequence. In situations ③ and ④, the earlier instruction is sent through first, followed by the later instruction.
The address that is entered in the memory control unit has an initial address, an address modification method, and an increment. Thus, it is necessary to compare not only the initial address, but the modification range as well. This can be done with two address subtractors. The first compares the starting addresses and the second compares the starting address plus the modification range (DE + 16 ΔZ compared with DH + 16 ΔH).

In internal storage address collision comparison, the initial internal storage address DZ in the transfer to stack buffer (J_{ZDH}) instruction read out of the instruction stack is compared with the initial internal storage address DH of the data send instruction in stations H_0 and H_1. The comparison is carried out by using the first adder in pipeline station 0 as $DZ_{(0-18)} - DH_{(0-18)}$: when $DZ_{(0-18)} = DH_{(0-18)}$ a collision is recognized. If $DE_{(0-18)} \neq DH_{(0-18)}$, to simplify processing we distinguish two more situations:

(1) Within the same vector loop $\overset{*}{[}\ \overset{*}{]}$ or the program segment $T_{non} = 1$ or the instruction to be compared is a scalar and the corresponding stations H_0 and H_1 also contain scalar instructions. When $DZ_{(0-18)} = DH_{(0-18)}$ this is regarded as a collision, and if they are not completely equal the programmer must assure that there is no collision.

(2) Collision processing of instructions not in the same $\overset{*}{[}\ \overset{*}{]}$. When $DZ_{(0-18)} \neq DH_{(0-18)}$, a check is made to see whether the modified address

has a collision. It must also be borne in mind that the vector displacement (ΔZ, ΔH) or indirect control vector (QZ, QH) method can be used to modify the address, and the displacement may be positive or negative. In this way there can be eight possible combinations. The second-level adder in pipeline station 0 carries out comparison of the range, and identification of the result yields the condition for no collision. In addition, the internal storage space (520,000 words) is divided into 8 blocks, each containing 64,000 words. A simplified condition for no collision which assures correctness is the prescription that the address space displacement for 16 ΔZ, 16 ΔH, (QZ) and (QH) must not exceed 64,000. Collision comparisons may be made within the 64,000-word range, and cases in excess of 64,000 are treated as collisions.

When a new data fetch instruction and the data send instruction in stations H_0 and H_1 are not in collision, the instructions not in collision can be moved up in the queue and executed. These operations require a collision comparison time of one to two beats, and comparison with either H_0 or H_1 requires that the comparison be carried out by the second-level adder within one beat. Therefore the adder structure and the condition formulation must be meticulously designed in order to assure correct, rapid completion of collision comparison.

Figure 4 presents a block diagram of the collision comparision station for data fetch and send described above.

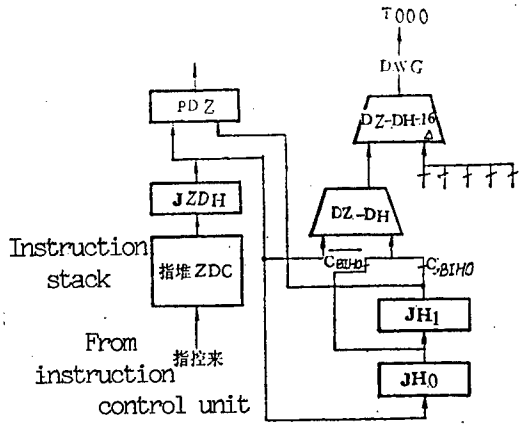


Figure 4. Collision Comparison Station for Data Fetch and Store Address

In addition to collision comparison for internal storage addresses, there are also problems involving comparison of Q and h. The 757 has send Q and use Q instructions and send h and use h instructions. If the Q to be used is in collision with the previous instruction requiring sending of Q which is present in station H, then the later instruction must wait: otherwise it can be moved up and executed ahead of sequence. The same applies to h.

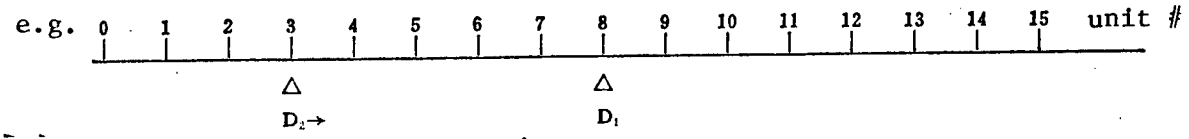
III. Altered Readout

The 757 vector machine uses 16 storage units, each with a capacity of 32,000 16-bit words, in modulo-16 operation. If the address modification increment for the 16 components of an instruction is odd, then in modulo-16 operation it is possible that none of the units will be in collision. But two vectors will not necessarily be continuous. If the former instruction fetches $\vec{A}$ (vector A) and the later instruction fetches $\vec{B}$, the relationship between these two vectors may be of several types.

If vector A is fetched from units 0, 1, 2, ..., 15, and the fetching of $\vec{B}$ can begin from unit 1 or from unit 8 or 15, the memory cycle is 15 beats (1.5 μsec), so that in terms of the overall probability of numbers of memory accesses, waits of 0, 1, 2, ..., 8, ..., 14 beats resulting from collisions may arise. If the probabilities are equal, then the number of waiting beats on the average will be

$$\frac{0 + 0 + 1 + 2 + \ldots + 8 \ldots + 14}{16} = 6.56 \text{ beats.}$$

The average waiting time of 6.56 beats resulting from memory unit collisions in fetch operations is quite considerable. Therefore, the altered readout approach was designed for the memory control unit. If the vector numbers of the preceding and following instruction are 16 and the increment is always (BZ) = 1, fetching of the following instruction begins at the unit next after the one in which the fetch of the previous instruction ends, and after 16 components have been fetched the vector is shifted to restore the correct form. In this way no waiting is required.



D_1(unit number address) $\geq D_2$(unit number address)+1

where D_1 indicates the number of the unit in which the second of the instructions begins; and D_2 is the number of the unit in which fetching of the first of the instructions ends.

Reading out of this instruction should actually begin from unit 8, but with altered readout it begins with the unit immediately after the one at which

readout of the preceding vector concludes. In the example above it begins with unit 4, and it can be seen from the table that starting at unit 4, 1 should be added to the unit address, while from unit 8 on, 1 should be subtracted from the address. Finally, because the vector position is not correct, it must be shifted to return the sequence to normal. The method is to use the condition that the unit numbers of D_2 and D_1 are equal to make the operation control unit start adding 1 to the cyclic 4-bit counter until the vector is completed. The final number of shifts should be equal to the complement of the 4-bit counter in question: in the example above, the shift is $(1100)_{comp}$ = 0100, i.e. 4, so that unit 8 is shifted to the beginning of the vector.

Normal readout			Altered readout		Cyclic counter
Cell address	Unit number address		Cell address	Unit number address	
100……0	1000	+1	100……01	0100	0000
100……0	1001		100……01	0101	0000
100……0	1010		100……01	0110	0000
100……0	1011		100……01	0111	0000
100……0	1100	−1	100……0	1000	0001
100……0	1101		100……0	1001	0010
100……0	1110		100……0	1010	0011
100……0	1111		100……0	1011	0100
100……01	0000		100……0	1100	0101
100……01	0001		100……0	1101	0110
100……01	0010		100……0	1110	0111
100……01	0011		100……0	1111	1000
100……01	0100		100……01	0000	1001
100……01	0101		100……01	0001	1010
100……01	0110		100……01	0010	1011
100……01	0111		100……01	0011	1100

It is evident from the foregoing that while altered readout adds a certain amount of hardware and complexity from the control standpoint, it has many advantages and it decreases the waiting time resulting from collision between units, thus increasing machine efficiency.

IV. Fault Diagnosis, Storage Odd-Weight-Code Checks, Instruction-Level Retry and Station-Level Double-Calculation

The 757 vector machine memory control unit has 31 error diagnosis points; at 25 of them, the machine is immediately stopped if an error occurs. The fault locations are kept within a small range in order to help with rapid discovery and elimination. The memory uses odd-weight code detection: it can correct single-bit errors and detect multibit errors. Because the memory has a relatively large number of units, the memory controller provides a common odd-weight code formation process and detection and correction circuits, which greatly decreases the amount of hardware needed.

When a datum read out of internal storage and received by the instruction controller contains a single-bit error, the error correction circuit corrects the error, records the number of the unit in which the fault occurred, and sends an access internal 2 (FNC II) code to rewrite the erroneous code within that unit. An extended read-write cycle is used to carry out the rewrite in the unit where the error occurred. New access can be made to that unit only after the rewrite is carried out and the memory control unit unlocks it. The objective of rewriting an erroneous entry in internal storage is to decrease the probability of 2-bit errors and to increase machine reliability.

Internal storage addresses, numbers read out of internal storage and numbers to be read into it are subjected to parity checks in the individual units, and if an error is discovered then the unit in question reports it to the memory control unit.

Addresses and data sent by disk or peripheral processor to the memory control unit use a common channel and are subjected to parity checks. If the memory controller finds an error in one of these data or addresses, it sends an error signal to the peripheral processor or disk. The peripheral processor or disk determines whether the error occurred in the sending of a datum or an address. In addition, after the memory control unit corrects single-bit errors read out from internal storage it sends them to a disk or peripheral processor. The memory control unit is also capable of timely detection of double errors in readouts; it then sends a double error marker to the disk or peripheral processor.

If a fault or error occurs in stations 1, 2, or 3 of the memory control unit pipeline, the memory control unit can carry out instruction-level retry, because the instruction in question has not yet been discarded from the memory control unit's instruction stack. If the detection point in station 1, 2, or 3 detects an error, it immediately stops the machine, and instruction control unit T_1 (unified numbering register 1) indicates a fault at the third bit in the memory control unit and turns control over to the general interrupt controller, then moves to the start of the diagnostic program. The memory controller's current state is recovered under the control of the diagnosis and recompute problem and analysis and processing are carried out. Because the lengths of time for which the various control units stop the machine are different, in order to assure that the on-the-spot memory control unit stops the machine in the same beat as the fault appears, while the instruction control unit and the operation control unit need only stop the machine in the next beat, start-stop processing is also necessary. Thus the instructions must be cleared from stations 4, 5, and 6 of the memory control unit pipeline and some registers such as the look-ahead and look-behind registers must be set in the state before the instruction in question was executed, after which the diagnosis program initiates retry by the memory control unit, and the subsequent retry process is performed by hardware. The retry process depends on the nature of the instructions at stations 1, 2, and 3; starting with station 3, every serial instruction fetched from the instruction stacks for stations 1, 2, and 3 is retried.

If there is an external disk channel instruction in station 1, 2 or 3, since such instructions cannot be retried, the only course is to regard this external channel exchange with the disk unit as failed. When the retriable instructions for stations 1, 2, and 3 are retried, if an error does not arise, the "retry successful" signal is sent; this signal also requires that the memory control unit be stopped. After the diagnosis and recompute program detects the "retry successful" signal it makes a record and again starts the mainframe, and processing of the original program resumes. In the case of some jitter-type faults, retry can avoid change of path and stoppage of the machine, thus increasing machine availability and efficiency.

The 757's memory control unit also has station-level double-calculation. There are two types of station-level double-calculations. One is when a random jitter fault occurs in the station 0 collision comparator. This is reported to the general control and diagnosis program; after a program control delay of 200 msec the mainframe is restarted for reexecution, and if the fault is removed, operation may continue from the restart. In the other station-level double-calculation, a single error stops the station double-calculation. The memory control unit is provided with the C_{DCT} (single-error stop) flip-flop: when C_{DCT} is in the zero state the memory control unit can automatically correct a single-bit error in a datum read out of internal storage. When C_{DCT} is in the 1 state, if a 1-bit error occurs in the internal storage readout, the memory control unit must stop the machine (SMDCO). Similarly, bit 3 of T_1 reports to the general diagnosis controller, which switches to the diagnosis and double-calculation program. This program recovers and preserves certain error locations, finds the instruction that led to the error, and the number of the internal storage unit, after which the diagnostic program fans in the instruction to stations 3 and 5 and reads out an all-ones code and an all-zeroes code seven times from the unit which made the error. If an additional error does not occur during these seven readouts, then the memory control unit condition is restored, the program is used to correct the contents of the erroneous memory, and execution of the program is resumed. But if an error is still found during the seven readouts, this indicates a hard fault and the diagnostic program changes the memory unit status word and cuts out the malfunctioning unit, moves all of its contents to a backup unit, then restarts the mainframe and resumes execution of the program.

V. Memory Unit Switchover, Disconnection, and Address Mapping

The 757 vector machine's memory consists of 16 units and 2 backup units, each with a capacity of 32,000 words, or a total of 16 x 32,768 = 524,288 words. In general, at full capacity the machine operates modulo 16, and when one or two units are malfunctioning they are disconnected and replaced by backup units. Thus the unit can continue to operate modulo 16 with no loss of capacity. If a memory unit malfunctions while the program is being run, read and write are performed with a backup unit designated by the unit status word.

The internal storage into four major quadrants in terms of the highest address 01 bit, and the unit number is expressed horizontally from left to right. Accordingly, different lines express different unit addresses.

When there are more than two malfunctioning units, there are not enough backup units to serve as replacements, and it is necessary to disconnect some of the units. To assure continuity of addresses, at least four units are disconnected each time. If the number of malfunctioning units increases further, a maximum of eight can be disconnected, but they can only be within the same M8.

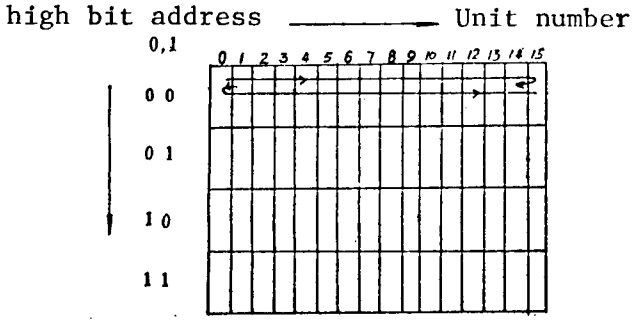


Figure 5. Modulo-16 True Address and Logical Address Operation

The machine can operate modulo 16 (M16), modulo 4 + modulo 8 (M4 + M8), or modulo 8 (M8).

In Modulo 4 + modulo 8,

Amputated units	Unit numbers in M4	Unit numbers in M8
QT_{0-3}(0-3)	4-7	8-15
QT_{4-7}(4-7)	0-3	8-15
QT_{8-11}(8-11)	12-15	0-7
QT_{12-15}(12-15)	8-11	0-7

In modulo 8 operation (number of malfunctioning units greater than 4),

Amputated units	Unit numbers in M8
QT_{0-3}, QT_{4-7}	8-15
QT_{8-11}, QT_{12-15}	0-7

When 4 units are disconnected, the internal storage capacity is decreased by 4 x 32,768 words. Because the operating system is stored in the last memory area, we regard the decrease as being the first one-fourth of the area. Only half of the first 260,000 units can be used, i.e., only 130,000, while the last 260,000 is entirely available. For example, when units 0-3 are disconnected, the correct addresses used by the first 130,000 units' cells are in units 4-7 in modulo 4 operation, while the last 260,000 are in units 8-15 in modulo 8 operation, as shown in Figures 6 and 7 (in Figure 7 units 0-3 are disconnected, units 4-7 are operating modulo 4, and units 8-15 are operating modulo 8).

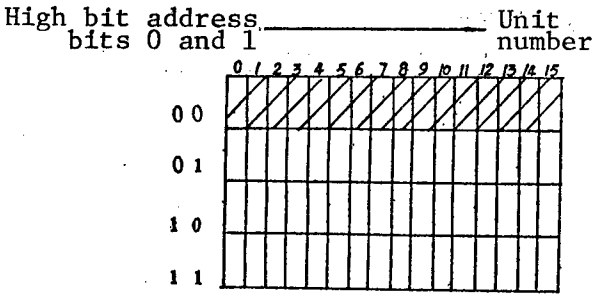


Figure 6. Logical Addresses in M4 + M8 Operation

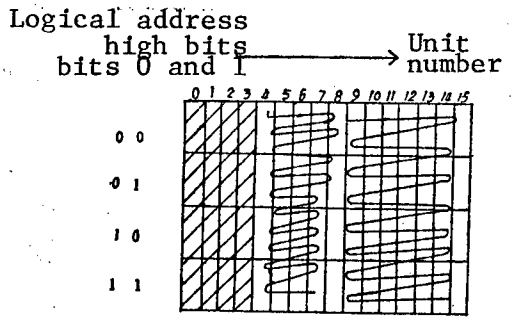


Figure 7. True Addresses in M4 + M8 Operation

When eight units are disconnected, the units in question are Nos 0-7, the capacity is decreased by 8 x 32,768 $\cong$ 260,000. Only the final 260,000 logical address remain, and when the units 0-7 are disconnected the modified true addresses must be stored in units 8-15 operating modulo 8, as shown in Figures 8 and 9.

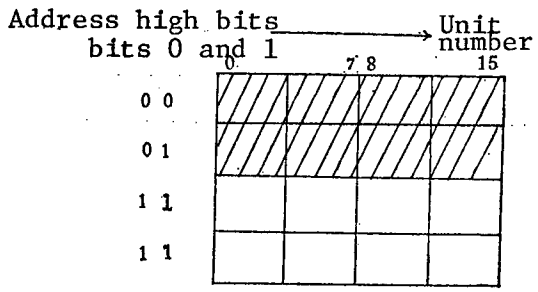


Figure 8. Logical Address in M8 Operation

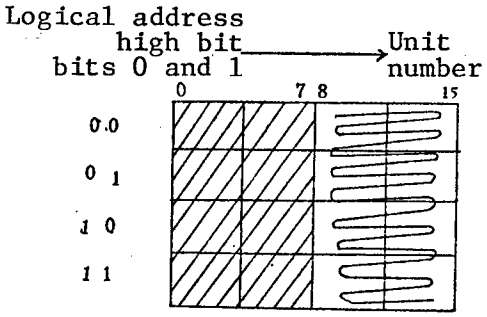


Figure 9. True Address in M8 (Units 0-7 disconnected)

When a unit is disconnected, the module number, the true addresses, the unit numbers and the address mapping are all implemented through memory control unit hardware. The amount of hardware required is not large, but it can increase machine reliability and memory unit availability. In general, with 16 memory units, an average of 1 unit a day malfunctions, and if it is replaced by a backup unit can be repaired within a day, the machine's speed and efficiency will not be degraded. Otherwise the unit must be disconnected and machine efficiency is temporarily degraded, but the machine can still operate normally.

To summarize, the 757 vector machine's memory control unit sees time-shared queuing, an overlapped pipeline, and bus operation; under present conditions these features, together with those described above, save hardware, increase speed and reliability, and have achieved rather good performance.

The 757's memory control unit was developed by collective effort; participants in the work included Comrades Yang Shufan [2799 2885 3058], Liu Pixuen [0491 0012 4821], and Wang Maojie [3769 5399 6738].

8480/9365
CSO: 4008/199

DESIGN CHARACTERISTICS OF 757 VECTOR MACHINE'S INSTRUCTION CONTROL UNIT

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 34-38

[Article by Li Shuyi [2621 2885 6318] and Luo Yinfang [5012 6892 5364], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] The 757 machine is a large-scale high-speed computer system consisting of a peripheral processor and a mainframe (vector machine). This machine is made entirely with Chinese-made components and devices; the vector machine uses high-speed ECL [emitter coupled logic] small-scale integration [SSI], and some medium-scale integration [MSI] circuitry. The internal storage uses magnetic core memory. From the choice of the overall design approach through the logic design and engineering implementation, assuring machine reliability and operating speed was constantly a major concern. Several problems related to the design of the instruction control unit are discussed as follows.

I. Choice of the Instruction Control Unit's Mode of Operation

The 757 machine is a single-instruction-flow, multiple-data, serial-data-feed and serial-data-processing pipelined computer. The instruction flow, from the internal storage (NC) through the memory control unit (CK), the instruction control unit (ZK) and the operation controller and ALU [arithmetic-logic unit] (YI and YS), performs the process of instruction fetch, interpretation, and execution. One vector instruction can process a maximum of 16 numbers; when performing vector operations, 10 million results can be obtained every second; in scalar operations, between 2.5 and 3 million instructions can be processed per second.

The task of the instruction control unit is to fetch instructions from internal storage via the memory control unit, to decode operation instructions and compute their data addresses, and to decode and execute control instructions. The time (P beats per instruction) required for the instruction control unit to feed one instruction to the memory control unit and to the operation control unit and ALU may be in one of the following three relationships to the specified vector length W in the instruction:

$$P = W \quad (1)$$
$$P < W \quad (2)$$
$$P > W \quad (3)$$

The choice of P determines the mode of operation of the instruction control unit. If P = 1, it is best to use the pipelined mode of operation, with one instruction fed to the two other control units during each beat. When P < W, the analysis speed in the instruction control unit is faster than the instruction execution speed of the memory control unit and the operation control unit-ALU. At this point, the memory control unit and operation control unit can be provided with buffers for temporary storage of instructions which have been interpreted in advance by the instruction control unit, thus compensating for the fact that instructions cannot be fed to them in time because of blockage of the instruction control unit pipeline. If $P \geqslant 2$, it is best to use the beat control method, whose logic structure is simpler than that of pipelining. When P and S satisfy equations (1) and (2), the machine's operation is as described above. When P and S satisfy equation (3), the instruction control unit will be slower than the other control units and there will be a speed mismatch in the system. Many factors affect the magnitude of P; the principal ones are the vector operation speed V_c, the scalar operation speed V_b, and the vector length in the instruction W:

$$P = f(V_c, V_b, W) \tag{4}$$

In the great majority of the 757's vector instructions, W = 16; when processing vector instructions, provided that P < 16 (which is very easy to achieve), the speed with which the instruction control unit interprets the instructions satisfies the requirements. Therefore, for a given W, V_b is the main factor in determining P.

Below, we select P for specified values of V_b. Assuming a machine cycle of T = 100 ns, when $V_b = 250 \times 10^4$ and 300×10^4 instructions/second, the values of P will be

$$P^* = \frac{1}{V_b \times T} = \frac{10^9}{250 \times 10^4 \times 100} = 4 \text{ beats/instruction}$$
$$P^* = \frac{1}{V_b \times T} = \frac{10^9}{300 \times 10^4 \times 100} \simeq 3.3 \text{ beats/instruction} \tag{5}$$

If we choose $P = \lfloor P^* \rfloor = 3$, we obtain $V_b = 333 \times 10^4$ instructions/second.

Figure 1 is an instruction processing flowchart for P = W = 3. In this figure, F is the instruction fetch period, T is the time during which the instruction control unit is interpreting the instruction (includes instruction decoding and computation of the data address or execution), L is the time during which the instruction is being loaded, and E is the time during which the instruction is being executed. It is evident from the figure that once the machine has executed two instructions the pipeline is full and thereafter one result is obtained per beat. Naturally, the real instruction flow in the machine will be much more complicated than that shown in the figure.

Based on the above analysis, a beat operation mode in which the instruction control unit fetches an average of one instruction every three beats was adopted; typical operation is shown in Figure 2. M_0-M_2 are the beat pulses,

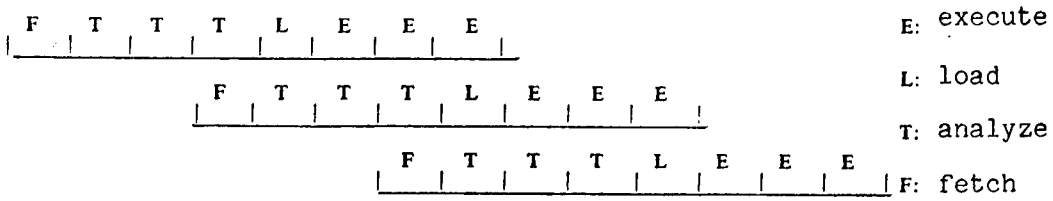


Figure 1. Instruction Processing Flowchart for P = W = 3

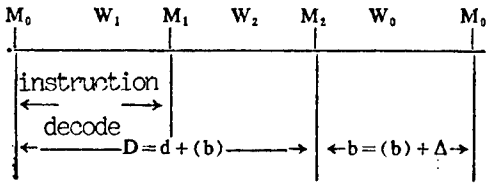


Figure 2. The Instruction Analysis

and W_0-W_2 are the beat control voltages. At M_0 the instruction interpreter loads an instruction from buffer ZH; during W_1 it decodes the instruction and begins to compute the data address, and at M_2 it obtains the logical address of the operand D = d + (b). The next operand address, b = (b) + Δ, is calculated during W_0, and preparations are simultaneously made to fetch the next instruction. In the data address computation formula, d is the immediate operand from the instruction, and b represents the index register. The data address increment is $\Delta = \ell(bz)$, where ℓ is the length of the vector, and bz is the increment storage location.

II. Design of the Instruction Dispatching Unit

The 757 vector machine uses magnetic core storage, which has a rather long read-write cycle; when modular addressing is used, if there is no access collision one unit of information is provided per beat. In order to decrease the demands upon internal storage and to assure that the ALU can process 10 million pieces of data per second, the instruction control is provided with instruction buffer storage ZH. When the ALU is not accessing internal storage, the instruction control unit can fetch additional instructions and store them temporarily in order to assure that when the ALU accesses memory, ZK [the instruction control unit] still will be able to provide a continuous flow of instructions and keep the pipeline operating smoothly. In designing the instruction dispatching unit, we were concerned primarily with the following two problems.

1. Choice of the Capacity of ZH [Instruction Buffer Storage]

In general, the greater the capacity of ZH, the smaller the demands on internal storage resulting from instruction dispatching. Naturally it would be best if the capacity of ZH were so great that it could hold an entire job program, but this is not feasible. In order to limit the amount of hardware used while not degrading machine efficiency, we had to make a statistical analysis of several problems, focusing primarily on branching.

Suppose that a branch instruction is stored in unit A and the target address of the branch is D. Then the dynamic statistics for this branch are as follows:

Let the branch distance be $\Delta = |D - A|$;
and suppose that the number of branches with $\Delta = 1$ is n_1;
the number of branches with $\Delta = 2$ is n_2;
⋮
and the number of branches with $\Delta = m$ is n_m,

then the percentage of branches in which the distance is a given length is

$$Si = \frac{n_i}{\sum_{j=1}^{m} n_j} \tag{6}$$

and

$$\sum_{i=1}^{m} S_i = 1 \tag{7}$$

We can use our computation results to plot S against Δ, as shown in Figure 3.

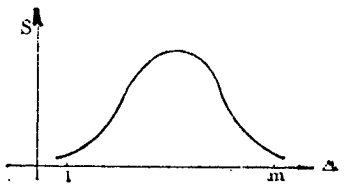


Figure 3. Plot of S vs Δ

For example, the statistical results for a large problem with PAF computation were

$$\begin{aligned} &\text{when} \quad \Delta = 32, \quad \sum_{i=1}^{32} Si = 50 \text{ percent} \\ &\text{when} \quad \Delta = 64, \quad \sum_{i=1}^{64} Si = 96 \text{ percent} \end{aligned} \tag{9}$$

In other words, when the capacity of ZH is 32 units, 50 percent of the branch instructions will branch to an address outside of ZH. If ZH has a capacity of 64 units, then only 4 percent of the branch instructions will go outside ZH's address space. This provides numerical data for deciding on a capacity for the instruction buffer.

2. Instruction Dispatching Strategy

In order to make full use of the storage space in ZH, we used several flexible instruction dispatching strategies in the instruction control unit. The principles on which these strategies are based were as follows.

(1) If there is vacant space in the instruction buffer storage, new instructions are dispatched into the vacant space. Because ZH has a large capacity, instructions fetched to the instruction register can be fetched from ZH.

(2) The space in ZH occupied by instructions which have already been executed is regarded as vacant, which assures that the next instructions in a sequential program will be continuously dispatched to ZH.

(3) The flag in a branch or loop instruction can be used to "fix" a program segment in ZH to assure high-efficiency transfer of loop programs.

To implement the above principles, we provided the instruction control unit with lower boundary address register xd, upper boundary address register sd, and control flip-flop CZY. Register xd contains the current starting address in ZH, and sd holds the address at whch the current instruction dispatched to ZH is written (sd is incremented by 1 each time an instruction is dispatched). When CZY = 1, dispatching of instructions to ZH ceases.

Control over dispatching is exercised as follows:

(1) When the difference between sd and xd is less than 64, this indicates that ZH is not full, and instructions are dispatched into it.

(2) When the difference between instruction counter JSZ and xd is equal to or greater than 16, this indicates that at least 16 instructions that had been dispatched to ZH have been executed, so that there is room in ZH into which new instructions can be dispatched.

(3) When JSZ and sd are equal, this means that all of the instructions in ZH have been executed and more instructions can be dispatched to it. But under these conditions, if a loop termination instruction is the instruction currently being executed and the last pass through the loop has not yet been reached, because the instructions in ZH are a loop segment, no further instructions are dispatched to ZH.

(4) When the target address for initiating a job or a branch instruction is not stored in ZH, the contents of ZH are discarded and the starting address of the job or the target address is sent to xd and sd, while an instruction is simultaneously dispatched to ZH. In this case, when the first instruction is forwarded to ZH, it is simultaneously forwarded directly to the instruction register JZ for execution in order to speed up its interpretation and processing.

(5) When a loop opening instruction is being executed, if flag bit $T_2 = 1$, then the contents of JSZ is forwarded to xd. When a loop closing or branch

instruction is being executed, if flag bit $T_1 = 0$, then flip-flop CZY is set at 1 and dispatching of instructions is suspended. Flag bit T_2 can be used to "fix" a loop segment in ZH, while flag bit T_1 can be used to prevent instructions not currently being executed from being dispatched to ZH, which increases machine efficiency.

(6) When carrying out such instructions as transfer to the supervisor mode, setting of a switch, or end of computation, which change the machine state, the instructions previously dispatched to ZH are discarded and dispatching of instructions is begun again. Therefore, when such instructions are among those dispatched to ZH, CZY is set at 1 and the dispatching of new instructions to ZH is suspended.

III. Reliability Design of the Instruction Control Unit

The indicators of machine reliability include the following:

1. Average time of stable operation: for the 757's CPU, this is 50 hours.
2. Average machine utilization rate P:

$$P = \frac{\text{effective operating time}}{\text{effective operating time + maintenance time}} \tag{9}$$

Here by machine reliability, we mean the use of logic to decrease the machine repair time.

The faults which result in machine maintenance include solid faults and intermittent faults. A complete fault detection system includes fault notification, double calculation, and diagnosis. The objective of fault diagnosis is fault location, and it deals with solid faults. The objective of double calculation is fault-tolarant operation, and it deals with intermittent faults. The double calculation and diagnosis functions can improve machine reliability, availability, and serviceability (RAS). Below we discuss several topics related to fault detection hardware.

1. Set-up of Fault Detection Points

The fault detection points form a chained testing system; single faults and some multiple faults can be detected and announced during the current beat in any element of the instruction control unit. The fault detection algorithms differ from circuit to circuit.

(1) Registers are all provided with parity bits and parity test circuits.

(2) Parity prediction, setting of the parity bit, and parity testing are performed in the counters.

(3) Matching of duplicated devices is used in the address adder.

(4) Parity prediction and matching of duplicated devices are used in important control circuits.

2. Save the Fault State

When a fault message is emitted at a detection point, the fault state must be rigorously saved in order to allow the fault to be located. This requires that emission of the next pulse be stopped in the beat during which the fault is detected in order to prevent propagation of the fault.

Signal transmission in a logic circuit composed of D flip-flops can ultimately be simplified to transmission between a register, combinational logic circuitry, and another register as shown in Figure 4. The combinational circuitry in the figure includes the fault detection circuitry.

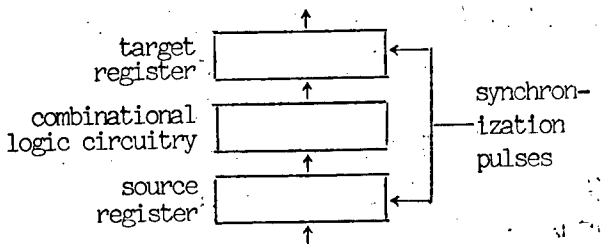


Figure 4. Signal Transmission Paths in Logic Circuitry Made Up of D Flip-Flops

In order to save the fault state, the following condition must be met:

$$T_s + T_c < T \tag{10}$$

where T_s is the time required for the signal to pass through the combinational logic circuitry, T_c is the time required for locking of the pulse, and T is the pulse cycle. Therefore, during the design process we imposed strict limitations on the lengths of all logic chains. In the debugging process we carried out an "alignment" of synchronizing pulses with reference to the actual chain length so that the instruction control unit could operate at a frequency 10 percent above the design value and the situation at the time of the fault could still be saved.

3. Recovery of the Fault Site

The instruction control unit is provided with a recovery register which can recover the conditions of the flags, registers and flip-flops and the important voltage levels at the time of the fault in order to allow the operator or the diagnostic program to locate the fault.

4. Double Calculation

The maintenance time in equation (9) includes the time needed to repair fixed faults and intermittent faults. Past experience indicates that in a large computer, intermittent faults are much more numerous (3:1) than permanent faults; their intermittent character makes them difficult to locate, which

presents great problems for maintenance and greatly lowers the computer's utilization rate P.

The basic concept of the double-calculation function is that the machine flow is temporarily stopped at the point of the failure; if the fault at this location is intermittent it may disappear spontaneously by the end of the pause, enabling the machine to continue operation. Double calculation is divided into program-level and instruction-level double calculation. The execution time in instruction-level double calculation is very short and is negligible in comparison with the maintenance time in solid faults; this fact greatly increases P.

The pipelines in the 757 machine are very long: when a fault occurs while the ALU is executing the K-th instruction, the instruction control unit may already be processing the K+7-th instruction. There may be several "destructive" instructions between the K-th and K+7-th instructions, and while the K-th instruction was being executed in the ALU, they will have already changed the contents of some registers or memory cells. Therefore it is pointless to have the instruction controller simply reload the K-th instruction and reexecute. For ease in selection of the point at which the double calculation begins, the 757 vector machine uses unified control by the double calculation software, with individual fault detection by the three control units at their respective interfaces, strict distinguishing of faults in different control units, and separate double calculation in each control unit. The instruction-level double calculation approach is used in the instruction control units. Below we discuss several problems related to double calculation.

1. Selection of Starting Point for Double-Calculation. The machine display of a fault site may be rather complicated, and if the fault site is used as the starting point for the double calculation, greater problems may result. A simple and effective method is to use the instruction in which the fault arose as the double calculation starting point. In this case it is only necessary to clear the instruction control unit and forward the instruction during which the fault occurred from the address specified in the double calculate instruction control register JSZ_1. The principal prerequisite for using the instruction during which the fault occurred as the double calculation starting point is that the program in internal storage be correct.

2. Distinguishing Double Calculable and Not-Double Calculable Faults. Because in certain instructions the contents of some register or memory cell may be altered at the end of a certain beat, if a fault occurs after this point to double-calculate is pointless. Therefore when designing the instruction control unit we made an analysis of the faults occurring at different beats in each instruction and determined which were double calculable faults and which were not; the relevant flags are either set or not set accordingly.

3. Choice of Double Calculation Control Process. When a double calculable fault has occurred, the instruction control unit must stop emitting pulses, save the current site, and notify the double calculation software of the

fault via the interrupt system so that the double calculation software can recover the current site, record it, clear the instruction control unit, dispatch the instruction to be double-calculated as specified by the contents of JSZ_1, and execute it. If no error occurs during execution of the double calculation of this instruction, only a "double calculation successful" interrupt signal is emitted, and the double calculation software is notified. If the double calculation is not successful, then the double-calculate software transfers control to the diagnostic software for fault diagnosis.

Both the structure cards and the control cards of the instruction control unit can send fault messages and diagnose and locate faults. If faults occur in important units on these cards, double calculation is still possible; and the extent of fault detection coverage is rather great. In a large computer, detection hardware generally accounts for about 25 to 30 percent of all hardware; it accounts for about 15 percent in the instruction control unit, and the amount of equipment used directly for double calculation is even smaller.

8480
CSO: 4008/200

LOGIC PARTITIONING OF 757 VECTOR COMPUTER INSTRUCTION CONTROL UNIT

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 39-43, 25

[Article by Luo Yinfang [5012 6892 5364] and Sun Shuzhen [1327 3219 3791], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] Logic partitioning of a computer constitutes the overall design activity in the computer engineering stage; it divides the entire machine's logic into levels and partitions it into small units which are easily laid out and replaced. The quality of logic partitioning affects machine speed and production cost, as well as reliability and maintainability.

The 757 vector machine is a large, high-speed pipelined computer. It is assembled from ECL small-scale integration [SSI] and some medium-scale integration [MSI] circuitry and its logic is partitioned in three levels: frame, board, and card. In the past, logic partitioning was pieced together based on the experience of the engineering and technical personnel and their familiarity with machine logic; there were no general rules on which to rely. This meant that in the case of the third logic level, and particularly in the case of the control cards, because of their logical complexity, chaotic arrangement and size and the fact that they had always required the most iterations and were the most time-consuming components in computer engineering, logic partitioning was a major factor affecting the quality of the computer. When partitioning the 757 vector machine's control cards, we produced a mathematical description of the logic partitioning on the basis of logic diagrams and bipartite graphs, and used network connection matrices and tables to partition the logic; this approach achieved good results, while not only decreasing the time spent and reducing the number of iterations, but also producing a rather rational partitioning and providing experience for the automation of partitioning.

I. Mathematical Description of Logic Partitioning

Logic partitioning begins with logic diagrams and engineering specifications. Once logic diagrams are expressed in terms of equivalent bipartite G graphs, the logic elements distinguished in the partitioning, such as boards and cards, must satisfy the G graph node union and intersection equation (1) in logical terms and also must satisfy the equation (2) regarding space for the elements and their internal connections in engineering terms:

$$\left.\begin{aligned}&\bigcup_{i=1}^{n} H_i = H\\ &H_i \cap H_j = \phi\end{aligned}\right\} \quad (\phi \text{ is the null set, } i \neq j) \tag{1}$$

$$\left.\begin{aligned}&K(H_i) = \sum_{a \in H_i} K(a) \leqslant K_{max}\\ &p(H_i) = \sum_{a \in H_i}^{p} p(a) - \sum d(s) - \sum (d(s)-1) \leqslant p_{max}\end{aligned}\right\} \tag{2}$$

H in equation (1) is the node set of the G graph, and H_i and H_j are two subsets of H. In the partitioning of levels 1 and 2, these subsets represent the frame and board. When we discuss the partitioning of the instruction control unit's control cards, H_i and H_j refer to cards. For example, the look-ahead station distributor diagrammed in Figure 1(a) is a logical network consisting of nine logic elements (a_1-a_9, of which five are flip-flops, three are gates and one is a half-adder), whose equivalent G graph, shown in Figure 1(b), includes nine circuit nodes formed by logic elements (a_1-a_9), nine boundary signal nodes (S_1-S_9) formed by networks, and four internal signal nodes (S_{10}-S_{13}). H in Figure 1 is the set of the a_i's and S_i's, while H_i is a set consisting of some of these nodes. Thus equation (1) includes two meanings: 1) the nodes in the G graph for the control logic of the instruction control unit must be the totality of the nodes on the control cards, and no logic must be omitted; and 2) the nodes on the control cards do not overlap, i.e., after partitioning, logic is not duplicated.

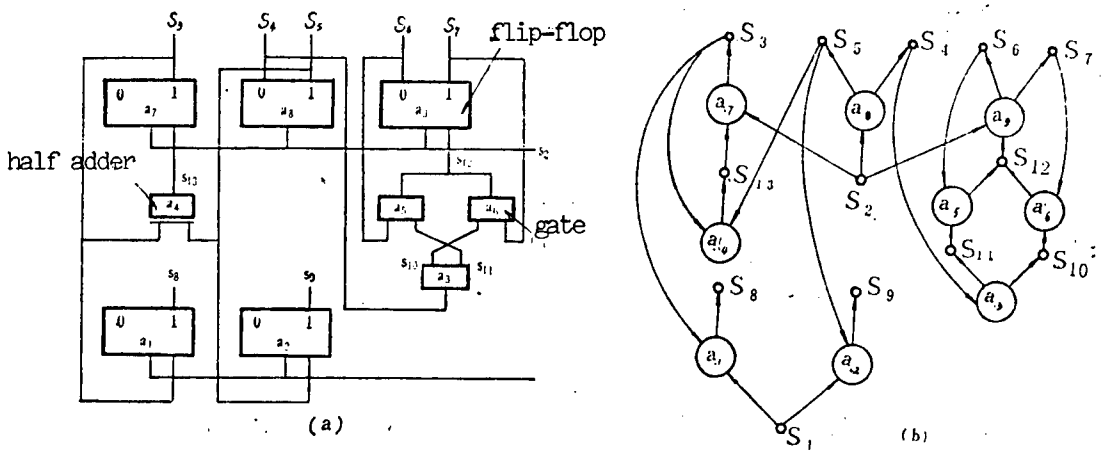


Figure 1. Distributor of Look-ahead Station

(a) Logic diagram;
(b) Logic diagram bipartite graph G

The first formula of equations (2) states that the number of [integrated circuit] packages partitioned onto a card must not exceed the permitted layout number K_{max}. K(a) in the equation denotes the number of packages used in logic element a_i. If the look-ahead station distributor shown in Figure 1

is partitioned onto a single card, and if one IC package represents one logic element, then the card will have nine IC's. The type A cards in the 757 vector machine can hold 78 IC's and the type B cards 48. Since the instruction control unit's control cards are type A cards, we choose $K_{max} = 78$. The second formula of equations (2) gives the card connection factor and states that the number of card pins used must not exceed the permitted number P_{max}. The first term in the equation gives the number of internal connections in the set of logic elements, the second term gives the number of connections between internal signal nodes, and the final term gives the number of connections of the set of boundary signal nodes. Because the number of external connecting lines in Figure 1 is 28, the number of internal signal node connections 9, and the number of boundary signal node connections 10, when the logic of Figure 1 is partitioned onto a single card, the number of pins used is 9 = 28 - 9 - 10. Since the instruction control unit is on a type A card, we have $P_{max} = 108$ lines. The maximum number of external connection lines for a type A card is $\sum_{a \in Hi}^{P} p(a) = 1100$ lines.

II. Criteria of the Feasibility of Partitioning of an Instruction Control Unit Control Card

Whether a partitioning of a card is feasible depends primarily on whether the number of IC packages used exceeds K_{max}, whether the number of pins used exceeds P_{max}, whether the logic chains on the card are within the specified speed range, whether the loads of the drivers on the cards are within the specified range, and whether the wiring on the card exceeds the specified number of connections. In general, if a card meets the above five specifications the partitioning is feasible. Naturally, a feasible partitioning is not necessarily optimal or even good. Therefore, to these five basic criteria we add several quality criteria for partitioning. Below we list the feasibility criteria for instruction control unit control cards.

1. The maximum number of IC packages K_{max} per card is 78.

2. The maximum number of pins per card P_{max} is 108.

3. The maximum number of transmission levels per logic chain is L = 23.5, including 17.5 logic levels and 6 levels worth of 3-meter transmission line delay.

4. The maximum number of connection wires per card is $\sum_{a \in Hi}^{P} p(a) = 1100$ lines.

5. The maximum load on the drivers is G = 8. If loads on another card are driven, they must all be limited onto the same card.

6. The gate-to-pin (or package-to-pin) ratio $\xi \geqslant 0.5$. This is an aggregate indicator of the space factor and the external connection factor for the cards. The greater ξ is, the better the card is partitioned.

7. The minimum number of IC packages per card $K_{min} = 20$. Because ξ is a ratio, it may be large without the cards being filled by IC's. Therefore,

a card can only be regarded as effectively partitioned when a minimum number of IC's per card is specified and ξ is large. But it must be borne in mind that because there are several gates (or logic units) per card, when the gate utilization factor is not high, even if ξ is large the real gate-to-pin ratio will still not be large.

8. The values of the elements A_{ij} of the A matrix in the initial state must be no greater than 20 in order to assure the quality of the card type subgroups.

9. One logic element must be and may only be partitioned onto a single card. This is in order to prevent duplications and omissions. When the nodes have different names, this requirement will naturally be satisfied during partitioning.

The partitioning feasibility criteria are the constraints on logic partitioning. In this sense, logic partitioning is a problem of finding an assembly algorithm which will satisfy the various constraints. If the requirements regarding card partitioning are extremely stringent, the feasibility criteria will of course be numerous, the assembly conditions will be complex and some of the factors will be mutually exclusive, so that logic partitioning will unavoidably be an iterative process involving continual revision.

Nine criteria have been specified for partitioning the instruction control unit's control cards. In order to simplify the problem and decrease the number of calculations, certain engineering requirements, such as a high IC utilization rate, the placement of IC's on the cards and the like were not taken into consideration.

III. Logic Functional Groups and Card Type Subgroups

1. Logic Functional Groups

The control logic of medium and small computers is relatively simple, and in order to decrease the number of card types, standard cards with relatively low values of ξ are used. But in large, high-speed computers this method will greatly increase the number of cards and may degrade machine speed. The control logic of the 757 vector machine's instruction control unit is implemented with specialized cards, of a large number of types; these make up nearly half the instruction control unit. Naturally, the attempt to achieve a high gate-to-pin ratio in logic partitioning becomes extremely important.

An effective means of increasing ξ is to group the logic elements in terms of logic functions and to place logic units belonging to the same functional group on the same card so that the networks connecting the logic elements can be contained within the card. The control logic functional groups in the instruction control unit were determined on the basis of the structural logic layout, the internal connections of the logic, and the requirement that the functional groups satisfy equation (2). The instruction control

unit's function includes more than 3,500 logic elements, which were grouped into 97 functional groups; these groups were then treated as the basic elements in card partitioning.

2. Card Type Subgroups

These groups are the nucleus of the card and are the functional groups that are distributed among the cards at the outset. The type subgroups, denoted a_i, are selected from the functional groups, and the remaining functional groups are denoted a_i'. The principles that were observed in selecting subgroups for the instruction control unit's control cards were as follows.

The a_i's are surrounded by a specific number of closely interrelated logic functional groups, and the logical relationships between the a_i's are as loose as possible; the number of a_i's may be large or small. Based on the structural logic layout of the instruction control unit, we chose 32 a_i's as card type subgroups and specified that one type subgroup would be used per card.

Once the partitioned logic was organized into functional groups or type subgroups, the logic partitioning was still described by equations (1) and (2), but the a's in the equations no longer denoted simply logic elements, but instead denoted logic functional groups or type subgroups; the nodes on the G graphs were also calculated in terms of functional groups.

IV. Partitioning of the Instruction Control Unit's Control Cards

When partitioning the instruction control unit's control cards, equation (1) was implemented in terms of a network connection matrix, and equation (2) in terms of a table of the a_i's and a logic distribution table for the b_i's and a_i's. As shown in Figure 2, the matrix and the tables are used to cross-check each other.

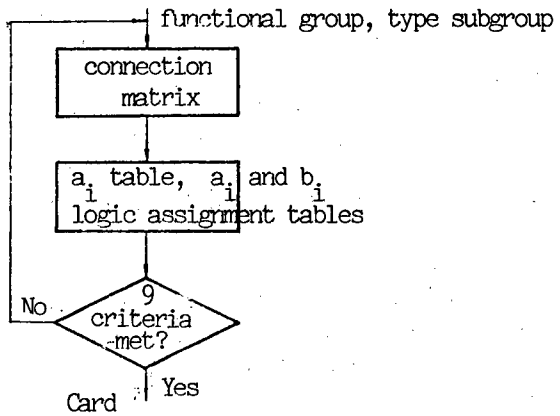


Figure 2. Relationship Between Matrix and Table

$$C = \begin{array}{c} a_1 \\ a_2 \\ \vdots \\ a_{32} \\ \cdots \\ a'_{33} \\ a'_{34} \\ \vdots \\ a'_{97} \end{array} \overset{\begin{array}{cccc} a_1 & a_2 & \cdots & a_{32} \end{array}}{\left(\begin{array}{c} A \\ \cdots\cdots\cdots\cdots \\ B \end{array} \right)}, \tag{3}$$

1. The Network Connection Matrix

The network connection matrix based on equation (1) describes the connections between different type subgroups and between type subgroups and functional groups. Matrix calculations can be used to decide the allocation of the functional groups and to test the quality of type subgroup selection. The form of the connection matrix used in partitioning the instruction control unit's control cards is shown in Figure 2.

The rows a_1, a_2, ..., a_{32} in the matrix represent the 32 type subgroups, while the columns represent the 32 type subgroups and the 65 functional groups that are to be allocated. The matrix includes submatrices A and B. Elements A_{ij} of matrix A express the connections between different functional groups, while the elements B_{ij} of matrix B express the connections between the functional groups which are to be allocated and the type subgroups. The connections between the a_i and a_j and between the a_i and a'_j (also denoted a_j in the table) are described by the following two equations:

$$c_{on}(a_i,a_j) = |B(a_i) \cap B(a_j)| \tag{4}$$

$$d_{is}(a_i,a_j) = |B(a_i) \cup B(a_j)| - c_{on}(a_i,a_j) \tag{5}$$

$$(i=1,2,\cdots,32, j=1,2,\cdots,97, i \neq j)$$

In the equation $B(a_i)$ are the boundary signal nodes of the type subgroups, while $B(a_j)$ are the boundary signal nodes of the functional groups. Equation (4) expresses the connections between the a_i's and a_j's, while equation (5) indicates the absence of connections between the a_i's and a_j's. If $d_{is}(a_i, a_j)$ is the result of the final card partitioning operation, this $d_{is}(a_i, a_j)$ is also the actual number of pins used on the card.

During the partitioning process, the matrices are continually changing, the a'_{33} $-a'_{97}$ are continually being combined and reduced, and the content of a_1-a_{33} is also continually changing. The initial state of matrix A expresses the interconnections between the type subgroups, and the sparseness of the matrix and the magnitudes of the A_{ij}'s indicate how well the type subgroups have been chosen. In other words, the sparser the matrix and the smaller the values of the A_{ij}'s, the smaller the number of logical relationships between the type subgroups and the better the choice of the logic subgroups. Ideally, all of the A_{ij}'s should be zero, but this is unattainable, and in the design process we specified $A_{ij} \leqslant 20$.

Matrix B expresses the relationships between the functional groups which are to be allocated and the type subgroups. Assuming that one type subgroup is placed on one card, matrix B expresses the card state during the logic partitioning process. When the elements of a row in the matrix are all zeroes, then there are no logical connections between the functional groups to be allocated and the card of which the type subgroups represented by the column are the nucleus. In other words, the partitioning of this card is complete and it can be removed from the matrix. When the elements of a line in the matrix are all zeroes, there are no logical connections between the functional groups represented by this line and any cards; but they may still have logical connections with other subgroups, so that when the row is removed from the matrix a new matrix can be formed, constituting a new card.

2. The a_i Card Characteristic Table and the b_i and a_i Logic Allocation Tables

The a_i card characteristic table (Table 1) records the characteristics of the card type subgroups and the characteristics of each card after the matrix computation. The characteristics recorded contain $d_{is}(a_i, a_j)$, K(a), L, and G. There are large numbers of L's and G's, and these need not all be included in the table. The L's and G's that are included in the table are the engineering logic levels of the external leads and the driver loads. Once card partitioning is completed, the final characteristics in the table should be tested in terms of ξ. If they meet the specifications, the card may be used, but if they do not, the card must be assembled and partitioned again. Generally, the type subgroups on such a card are changed into functional groups and a new selection must be made for partitioning. As a rule, K_{max}, K_{min}, the transmission speed and the load have been checked in the process of entering the K(a), L, and G.

When matrix B is used in laying out the logic, there are always rows and columns that are not all zeroes. When the row elements are not all zeroes (but more than one is), there is a logical connection between one of the functional groups to be assigned and several cards, and this functional group may be assigned to any of several cards. When the column elements are not all zeroes (but more than one is), there are logical connections between several functional groups and one card, and these functional groups may be combined on a single card. In the former case the logic is laid out by means of a b_i table in the latter case the a_i table is used.

The b_i logic allocation table (Table 2) records the initial number of external leads and of IC packages of the functional groups to be allocated, as well as the number of external leads and the maximum values of L and G after logic allocation to the a_i cards. This indicates to which a_i cards they are best assigned. The a_i logic assignment table determines which functional groups can be combined on a given a_i card. Because some of the functional groups that have been combined on a card have close connections with the a_i card and some functional groups have close connections between each other, the number of IC packages in these groups and their number of leads will differ. In this case, an assignment of functional groups to the a_i cards which maximizes ξ while the other characteristics meet the specifications should be chosen.

Table 1. a_i Table

K(a)	Type subgroup					First assembly					...
	Level, load			K(a)	dis	Level, load			K(a)	dis	
	L_1	G_1	...			L_1	G_1	...			
a_1											
a_2											
⋮											
a_{32}											

Table 2. b_i Logic Allocation Table

	K(a)	dis	a_1			a_2			...	a_{32}		
			dis	L	G	dis	L	G		dis	L	G
a'_{32}												
a'_{33}												
⋮												
a'_{97}												

Table 3. Logic Assignment Table for a_5

	a'_i assembly	K(a)	C_{on}	ais	ξ
1	5,6	15	0	×	×
	5,6,9	25	×	×	×
	5,6.9,18	34	×	×	×
2	5,9	17	12	6	2.84
	5,9.18	26	17	8	3.25
3	5.18	16	2	20	0.8
4	6,9	16	10	13	1.38
	6,9.18	18	16	8	3.37
5	6.18	27	0	0	0
6	9,18	19	5	10	1.9

The a_i logic assignment table is not large: Table 3 is a table that was used in partitioning the instruction control unit's control cards. It can be seen that when partitioning card a_5, groups a'_5, a'_6, a'_9, and a'_{18} were logically connected with this card. We now distinguish six circumstances in assembling the card. The number of IC packages, the connection characteristics (c_{on} and d_{is}), and the values of ξ are entered in the table. In the first assembly version, because there are no logical connections between a'_5 and a'_6 ($c_{on} = 0$), the two following assemblies (a'_5, a'_6, a'_9, and $a'_5a'_6a'_9a'_{18}$) are also not computed again (indicated by X's in the table); in the other five assembly versions the functional groups are all interconnected, and case 2 of assembly version 4 is the best; therefore we decided to place functional groups a'_6, a'_9, and a'_{18} on card a_5. The a_i table tells whether they meet the requirements regarding K_{max}, K_{min}, L, and G.

Because in matrix B, rows which are not all zeroes and columns which are not all zeroes occur simultaneously, the question of whether to use the b_i logic assignment table or the a_i logic assignment table is a matter to which particular care is paid. If the b_i table is used, functional groups with only

slight logical connections may occur together on the same card, while if the a_i logic assignment table is used, functional groups with only slight logical connections to the a_i cards may be included on the cards. Thus each approach has its advantages and disadvantages. Naturally the best method is to carry out several test computations, but in general it is better to use the b_i table, because it requires fewer computations.

C. Partitioning Flowchart for the Instruction Control Unit Control Cards

The instruction control unit control cards were partitioned in accordance with the flowchart in Figure 3. This chart is divided into three sections. In the first section, the G graph is drawn and the logic functional groups are partitioned in accordance with the logic diagrams and engineering specifications. In the second section the type subgroups are selected from the functional groups, A-matrix calculations are made, and the type subgroups which do not meet requirements ($A_{ij} \leqslant 20$) are rejected and are reclassified as functional groups. The third section includes B-matrix calculations and the making of the a_i tables and b_i and a_i logic assignment tables. Functional subgroups for which the matrix columns are all zeroes are reclassified as type subgroups, and the a_i cards are checked in terms of the partitioning feasibility criteria. The cards which do not meet these requirements must be repartitioned, while those which do meet the requirements are handled in one of two ways. The cards in which some columns are all zeroes are not included in the partitioning, while those in which the column elements are not all zeroes are subjected to further logic assignment.

The instruction control unit's control logic was ultimately partitioned onto 30 card types (i.e., onto some of the cards included in logic partitioning). The average number of IC packages per card type was over 55 and ξ was equal to or greater than 0.61, with a maximum of 1.14. In order to decrease the number of assembly iterations and the amount of computation, the logic that was left over at the end was assembled in ad hoc fashion on one card, which contained only 33 IC's and had a ξ value of only 0.31, indicating rather poor quality.

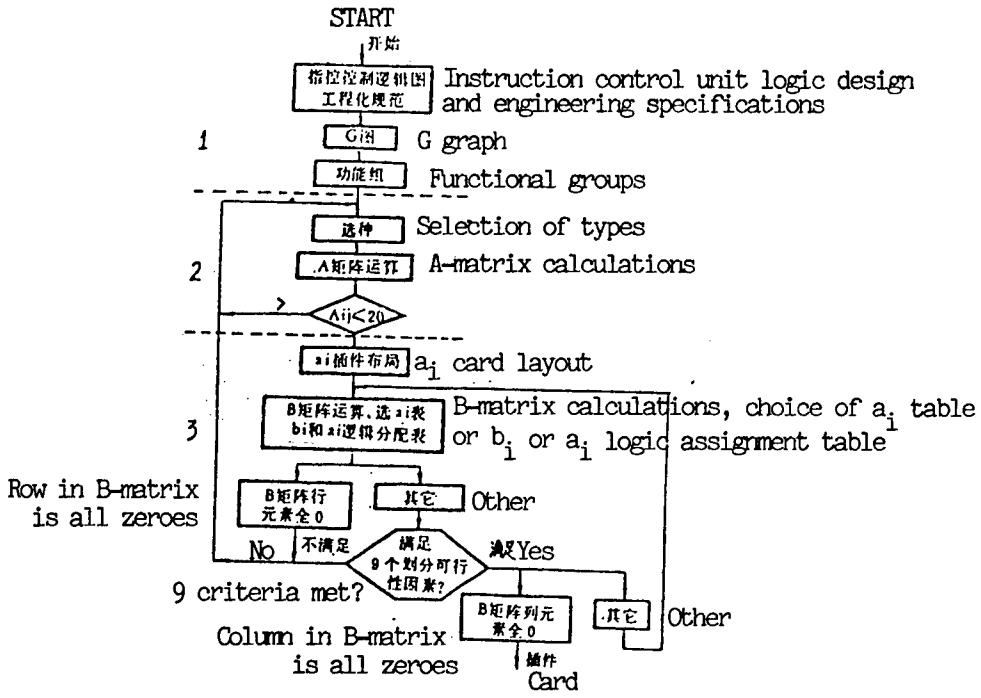


Figure 3. Flowchart for Partitioning of Control Cards of Instruction Control Unit

8480/9365
CSO: 4008/200

INTRODUCTION TO 757'S PERIPHERAL PROCESSOR

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 44-48

[Article by Mei Duolun [2734 1122 0243], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] What role does the 757's peripheral processor play in the system? How is it related to the vector processor? How do the two machines communicate? What are the characteristics of the way in which the peripheral processor is connected to the peripheral devices, and what is its machine language like? Below we summarize certain major topics regarding the peripheral processor.

I. Why Does the 757 Adopt the Peripheral Processor Scheme?

The vector processor is good at long vector calculations. Its efficiency can be effectively used in solving problems with a high degree of parallelism; but the efficiency of the vector processor will be lowered by problems with large numbers of scalar computations. But the computer's system programs involve principally scalar operations, and accordingly we considered using a peripheral processor to handle this time-consuming, repetitious work.

In addition, pipelining was used to implement high-speed vector computation. For the pipeline to operate smoothly, it is also desirable to have few interrupts and branch instructions. The system programs have a relatively large number of branch instructions, and accordingly it is advantageous to relegate these programs to a peripheral processor.

Having the peripheral devices directly controlled by the peripheral processor rather than connecting them directly to the vector processor decreases the number of interruptions of the latter, which also decreases the number of interruptions in the pipeline. In short, the peripheral processor approach was used in order to take advantage of the efficiency of the 757 vector processor.

II. Tasks of the Peripheral Processor

The peripheral processor, the principal link between the central and peripheral components of the 757 system, carries out system control functions.

Its specific tasks are: compilation of high-level language and assembly language, control of the entire system through the operating system (including task scheduling, resource management, I/O [input/output] data processing, I/O device control, man-machine interfacing and the like); and diagnosis of the vector processor.

III. Communication Between Machines

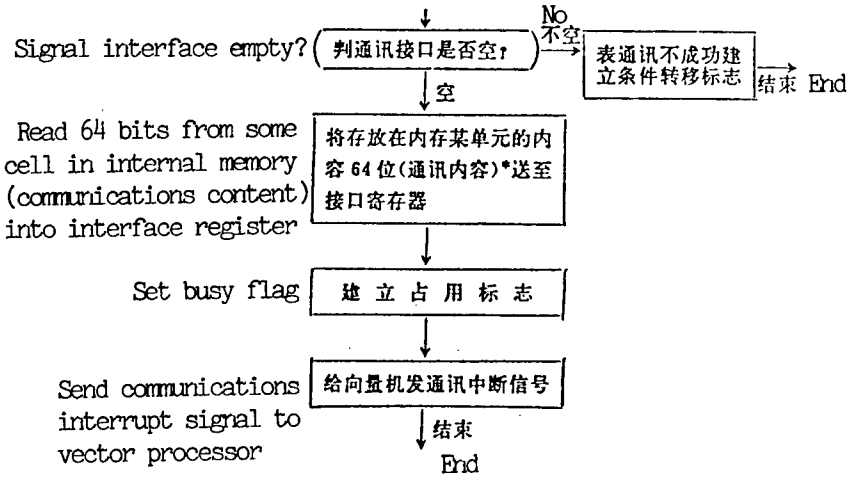
Since the peripheral processor, the vector processor, and the peripheral devices operate under the centralized control of the operating system, the machines must be able to communicate with each other and to exchange the information which they contain.

There are three specific ways in which the machines communicate information.

(1) Direct notification: When a fatal hardware fault develops in one machine, it immediately emits an emergency interrupt signal to the other machine.

(2) Communication: In order to meet the requirement for conversation between two machines, specialized interface equipment is provided. The vector and peripheral processor have their respective communication instructions, and when necessary each can use these instructions to initiate communication with the other.

The peripheral processor has two communication instructions. To initiate communication with the other machine, the "communication out" instruction is used. Its functioning is shown in the figure.



When the vector processor initiates communication with the peripheral processor, it also uses the interrupt method. When the peripheral processor responds and processes this interrupt, it must use the "communication in" instruction, whose function is to take the contents of the communications interface input register and transfer them to the internal storage cell

specified in the instruction. If they are correctly transmitted without error, the interface release signal is emitted, while if there is an error in the transmission, an interface error interrupt is sent. As can readily be imagined, the vector processor has two analogous instructions.

(3) Batch information exchange between the vector processor and the peripheral processor: The peripheral processor and vector processor do not share internal storage, and accordingly a channel for batch information transmission is required. For the peripheral processor, exchanging a batch of information with the vector processor is analogous to exchanging a batch of information with a peripheral device, and this interface is treated as one of 32 subchannels. The peripheral devices can be started or stopped only by the peripheral processor. The interface for batch transmission between these devices also can be started only by the peripheral processor.

In order to provide reserve capacity, the 757 system also can be equipped with an additional processor, called peripheral processor No 2, which is connected to the first peripheral processor just as the latter is connected to the vector processor. In addition, between the two peripheral processors there is a communication interface, a data transmission interface, and emergency notification, which we will not describe in detail here.

IV. Relationship of the Peripheral Processor to the Peripheral Devices

1. Survey of the Channels

All peripheral devices of the 757 system operate under the control of the peripheral processor. Of a total of 32 subchannels, 27 are in use and 5 are backup channels. In order to lighten the load on the peripheral processor and to allow it to handle information from the vector processor more rapidly, the eight subchannels for disk and tape units, an electrostatic printer, and a graphic display unit can also exchange data directly with the vector processor's internal store. In addition, they are divided into two direct paths to the vector processor's internal store. These direct-access channels also operate under the control of the peripheral processor. The peripheral processor is connected to the various peripheral device controllers by a channel controller, which controls the priorities of the various subchannels and the transmission of batches of data by each. When transmission of a batch of data has been completed, a completion interrupt is sent to the peripheral processor; thus the channel control word lacks a "zipper capability."

In addition, in order to meet speed requirements and provide direct access to the vector processor, the disk and tape units, the electrostatic printer, and the graphic display unit are also provided with controllers, which are generally called "disk-tape controllers." The data paths are shown in Figure 1.

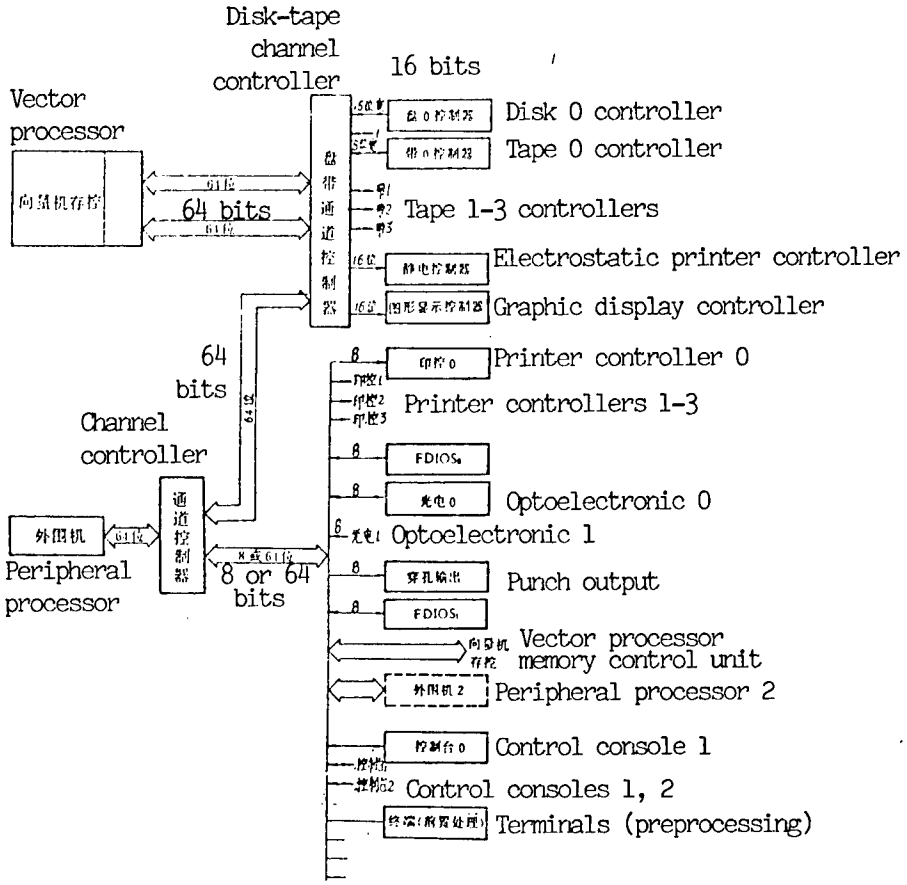


Figure 1. Schematic Diagram of the 757 Data Paths

2. How the Peripherals Are Used

No matter what the internal and external connections are, the supervisor program must send out a group of basic information for use of the peripheral devices. This information includes the internal storage starting address (for input, the storage of the information coming from outside begins at this address; for output, the information is sent starting with this address), the exchange length, the channel number, and the device number; in addition if a device operating code is to be exchanged with external storage, the external device address must also be specified. Different devices send this information in different forms.

The 757 peripheral processor lacks specialized peripheral device instructions; a register is provided for each category of information. Currently nine register addresses can be specified on each subchannel. Fewer are used in some subchannels. Then ordinary load and move instructions are used to access these registers, which is just as convenient as accessing an internal storage cell. Peripheral device operation is controlled by the contents of the registers. This type of connection and utilization provide flexibility and convenience for future expansion of the suite of peripherals.

V. Cross-Connected Channels

For the sake of reliability, the eight disk units are cross-connected into two subchannels. As the system diagram shows, if one channel malfunctions and cannot be used, all disk units can be accessed via the other subchannel, although the information transmission rate will naturally fall. The likelihood of both subchannels being inoperable simultaneously is small.

There are four magnetic tape drive subchannels: They are D_0, D_1, D_2, and D_3. D_0 and D_1 can be connected to eight tape drives between them; D_2 and D_3 also can be connected to eight tape drives between them, for the sake of reliability.

To enable the disk and tape units, electrostatic printer, and graphic display unit to directly exchange information with the vector processor's internal storage, two direct access channels are provided. Eight channels are used, with the odd-numbered and even-numbered subchannels forming groups. The provision of two direct-access channels is also for the sake of reliability: if one fails, the other can still be used to access all of the disk and tape drives. Under normal conditions the direct-access channels operate in parallel with high efficiency.

The direct access channels also operate under the control of the peripheral processor, which sends out peripheral device operating instructions, processes interrupt requests for the peripheral device controller and the channels, and recovers channel states.

Each of the eight subchannels connected to the two direct-access channels can individually exchange information with the peripheral processor's internal storage. But how is it possible to distinguish whether the peripheral device is exchanging information with the vector processor or the peripheral processor? Actually, the command word sent out by the peripheral processor contains a distinguishing flag.

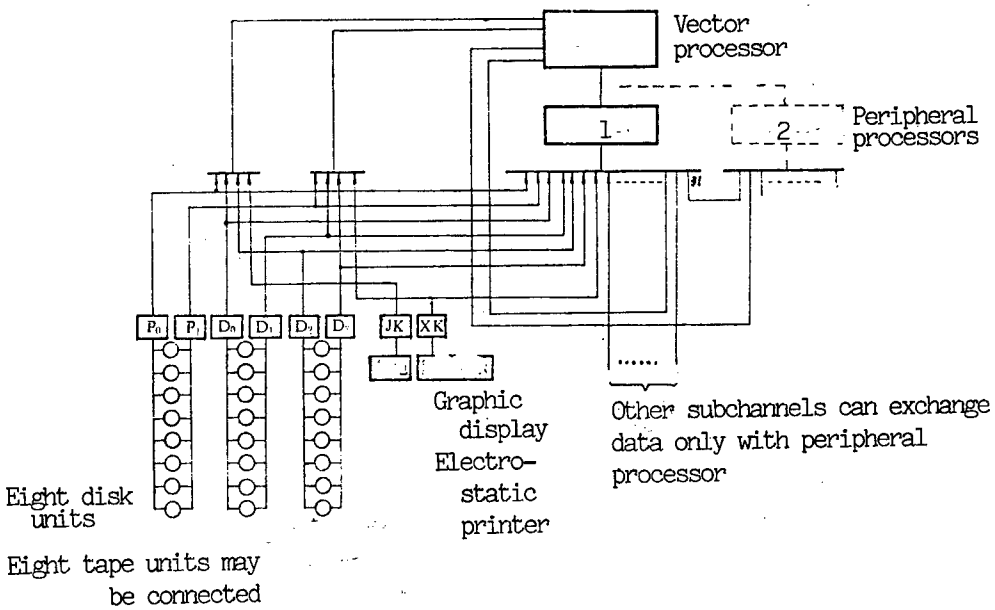


Figure 2. Block Diagram of the 757 System

VI. Capacity, Characteristics, and Reserve Capacity of the Peripheral Processor

The capacity of the peripheral processor is related to its tasks, the number of varieties of peripheral devices, and the capabilities of the operating system. Since this is the first design approach using a peripheral processor, there are as yet no statistics, so that we simply defined the capacity and specifications of the peripheral processor in terms of an analysis of likely situations. (1) The word length is 64 bits, using a fixed point binary complement system. Each memory cell contains two instructions with a single-address format. (2) The main frequency is 2.5 MHz and the average speed is 500,000 MIPS. (3) Storage: (a) 64,000 words of core storage with a cycle of 1.6 μs and with Hamming error detection. (b) Semiconductor storage of 64,512 words with a read time of 600 ns, using parity code. (c) Fast semiconductor storage, with a cycle time of 100 ns and a read time of 40 ns. The peripheral processor has a total of three semiconductor storages: the page table storage, the index storage, and the channel control storage.

Since it was not known whether this capacity would be sufficient, the possibility of installing a second peripheral processor was provided for in order to add extra capacity (see Figure 2).

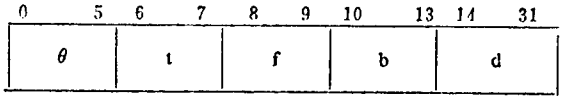
VII. Instruction Format and Instruction Set

Principal data formats: character strings, integers, various table formats.

Instruction format: it is desirable for the instruction format to be simple, complete, and convenient to use and to recall, because assembly language is still used to write other system programs.

1. Basic instruction type:

where θ is the opcode;



b is the index storage cell address (b = 1, 2, ..., 15), $(b_0) \equiv 0$,

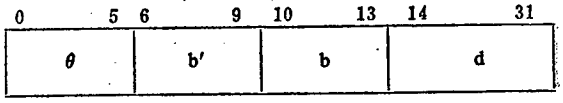
d is the formal address;

f is the formal address mode flag bit:

f = 0: (b) + d = D, where D is an 18-bit opcode (immediate addressing mode);
f = 1: (b) + d = D, where D is a logical address. In non-core-state programs, it is generally necessary to refer to the page lookup table to convert it to a real internal storage address;
f = 3: (b) + d = D, where D is also a logical address; and b + 2 → b, the self-incrementing mode; f = 3: (b) + d = D, where D is a logical address, and (b) − 2 → b, the self-decrementing mode.

t is a flag bit, whose meaning varies with the instruction type.

2. Indexed jump type: differs only slightly from type 1. The t and f fields are combined into a b' field. In an index type instruction, b' is the target address of the index unit, while in a branch instruction b can be used to save the return address or as a stack pointer indicating the saved return address:



Instruction set. The guiding concept was to keep it general with some special emphases, and when feasible to strengthen instruction capabilities as much as possible and to increase the execution efficiency of typical program segments. "Keeping it general" means use of ordinary general-purpose instructions, while "special emphases" refers to the fact that since the peripheral processor's primary task is to execute system programs, it has some special characteristics that distinguish it from ordinary large general-purpose computers. The system software characteristics are as follows.

It has different data formats. The peripheral processor functions primarily with integers and character strings and tables, while large general-purpose computers generally deal primarily with the floating point format. General-purpose computers solve numerical problems and generally work from equation to equation, using a relatively high proportion of arithmetic instructions, while system software generally moves from one table to another. Its processing activity characteristically involves table lookup, table creation, classification, and aggregation. For example, the operating system generally uses a variety of tables for the purpose of system management. When a new job is entered, for example, the operating system must fill out the "job backlog register." This is a table creation process. In high level scheduling procedures, selection of jobs from the job backlog register requires table lookup. As another example, when the user enters information as a file, the file management program sets up a file for the purpose, which is also table creation; and utilization of this file requires table lookup.

The compilation process is also a table lookup process, in which the original program can be regarded as a table composed of characters, and the various tables, such as the alteration pointer tables, name characteristic tables, and the like are used in various passes until a table in machine language, i.e., the object program, is produced. These requirements make the following features desirable.

(1) There should be flexible data load and store instructions. To provide for this requirement the 757 peripheral processor has: (a) standard load and store instructions for full word length data; (b) load and store of a bit or bit string in any position; (c) load and store of a field of any length within a memory cell; (d) use of immediate data. Computations with integers smaller than 2^{18} are more conveniently handled with immediate data. Its advantage is that it saves memory space and makes program inspection more informative. The immediate data format is generally used in address calculation and character recognition.

(2) Strengthening of instruction capabilities where possible in order to increase program execution efficiency. Naturally the basic capabilities should be drawn from large numbers of programs that are commonly run. For example, in the initial scan program of a compiler, to load each character, initially a "character-by-character load" subroutine was written. The way this subroutine worked was that each time it was entered, one character was loaded into the lower eight bits of the accumulator and a position was prepared for the next character by determining the memory cell in which it would be located and the byte number. This subroutine took about 10 instructions. But since it was a subroutine, on each entry it was necessary to execute two supplementary instructions (branch and return). The 757's periphral processor uses a machine instruction to replace the functions of this subroutine.

The "load character" instruction works as follows: Memory unit i is loaded from main memory; the j-th character enters the lower eight bits of the ALU, while at the same time a location is prepared for the next character to be loaded in accordance with the particular circumstances. There are generally four cases:

① Only one character is loaded, so that there is no need to prepare for the next character;

② The next character to be loaded is the same as the present character, i.e., there is no alternation in i and j (successor operation not necessary in instruction);

③ The next character is chosen in order, but it suffices to specify the successor operation. When t = 2, the successor operation specifies that J = J + 1. If J = 0 the character to be fetched is in the next memory cell, and in this case the hardware sets I = I + 1. I is the index element in the instruction. J is a three-bit byte register.

④ The character to be fetched next is fetched in reverse order, in which case the decrement flag in the instruction is used. Thus, after a specific character is fetched, if t = 3, then J = J - 1, and if, for example, J = 7, the next character to be fetched is in the preceding memory cell; in this case the change I = I - 1 is automatically made.

It should be made clear that before the character is fetched, the initial value is loaded into J; in sequential scanning, after this initial transfer all the operations are made automatically.

Corresponding to "load character" is "store character." Its function is to send the character in the accumulator to the j-th byte position in the i-th memory unit of internal storage, while all other character positions are unchanged. The alternations of i and j are the same as for "load character" and will not be described in detail here.

As another example, in order to replace units in page storage more rapidly, the peripheral processor is provided with a "store page" instruction. The

functions of the instruction are flexible, and four page units can be stored each time the instruction is executed.

(3) In order to solve synchronization and collision problems in the operating system, the P and V operations are used. These two operations can be implemented via two subroutines executed in the locked condition. Two instructions are used to carry them out in the peripheral processor: if the signal value is stored in D, the P operation is (D) - 1 → D, D → miscellaneous register, and when (D)* < 0, it produces an interrupt; the V operation is (D) + 1 → D, D → miscellaneous register, and when (D)* ⩽ 0, an interrupt is produced. These two instructions can increase the execution efficiency of the P and V operations.

In addition it is desirable that the processing of the branch subroutine be convenient and flexible. There are many ways of implementing the branch subroutine, but in general they use specialized instructions. In the peripheral processor, with ordinary conditional branches and unconditional branches the address of the next instruction can be saved as part of the operation. If the k-th instruction in the subroutine is

branch	b'	b	d

, the result of its execution is k + 1 → b', and the branch is (b) + d. Obviously, if it is not necessary to save k + 1, it suffices to write a zero in b'. But because there are only a few index units in hardware, this becomes inconvenient when the subroutine has many levels of nesting. In order to solve the problem of nesting in the next subroutine, two instructions, "branch" and "return," are provided. The idea is to arbitrarily open an arbitrary area in memory as a stack and use an index unit as the stack pointer. Then,

the function of the "branch" instruction is: $\begin{cases} k + 1 \rightarrow (b'),\ (b') + 2 \rightarrow b' \\ \text{branch to subroutine entrance} \end{cases}$

the function of the "return" instruction is: $\begin{cases} (b) - 2 \rightarrow b \\ ((b)) \rightarrow \text{program counter} \end{cases}$

These two instructions must be used as a pair. The advantage of this requirement is that they are easy to use and highly flexible, any area in memory may be used, and the stack size is flexible. This is easy to implement and no additional hardware is needed.

VIII. Address Protection

The peripheral processor uses the page protection method. Unified addressing is used for the core storage, semiconductor fixed storage, and miscellaneous registers; the entire address space is 128K words. The entire address space is now divided into 128 pages, each with 1,024 elements. The software manages them in accordance with system requirements and user requests for storage space allocation. The following are recorded for each page: the state (the program has three states: core, supervisor, and user) whose programs are being used, and whether just read or both read and write are enabled. A page table is implemented in semiconductor memory. The page table has 128 elements, each of 14 bits, including 2 parity bits.

When the logical addresses for instruction fetch or data load and store must be looked up in the table, the high-order seven bits (the logical page number) are used in the address to be accessed in the page store. The relevant element in the page store is accessed and a determination is made whether it has address protection. If access is permitted, the real page number stored in that page element and the low-order 11 bits of the page number are combined into a real address which is used to access the internal storage.

8480/9365
CSO: 4008/200

INFORMATION EXCHANGE BETWEEN 757'S VECTOR COMPUTER AND PERIPHERAL PROCESSOR AND DISK-TAPE CHANNELS

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 49-53

[Article by Liu Pixuan [0491 0012 4821], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] I. Overview

In the 757 large high-speed computer system, a two-machine architecture in which functions are divided between a vector processor and a peripheral processor is used in order to make full use of the high-speed vector machine's efficiency. The vector machine runs primarily applications programs and performs high-speed scientific calculations; the peripheral processor executes primarily system programs, runs the operating system and the compiler system, and controls the various peripheral devices.

During system operation, the vector processor and peripheral processor notify each other of situations, and interrogate and respond to each other; the vector processor must go through the peripheral processor to use the peripheral devices; batch information exchange between the vector processor and peripheral processor and man-machine communications must also be effected via the peripheral processor. The vector processor, peripheral processor, and peripheral devices make up the complete 757 computer system (see Figure 1).

In order to decrease the demands on the peripheral processor and increase information exchange speed, two high-speed direct digital channels are provided between the vector processor and the disk units, tape units, electrostatic printer, and graphic display unit so that disk and tape resources can be shared by the vector processor and the peripheral processor.

II. Information Exchange Between the Vector Machine and Peripheral Processor

There are three different connection types between the vector machine and the peripheral processor.

1. Direct Notification Mode: When an emergency situation develops in either of the two machines, the machine emits an interrupt message which is directly communicated at once to the other machine. After the other machine

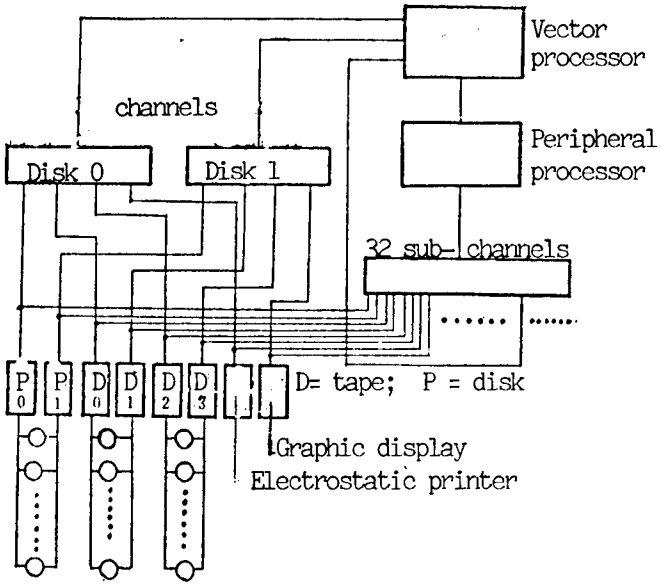


Figure 1. Block Diagram of 757 Computer System

has responded to the interrupt, it branches to the proper interrupt processing program. For example, when there is a power loss, machine stoppage or hardware fault in the peripheral processor, the peripheral processor emits a peripheral processor emergency signal to the vector processor, which is communicated to the latter directly, so that the vector processor ceases exchanging information with the peripheral processor. When a hardware fault occurs in any unit of the vector processor, it immediately emits a vector processor fatal hardware fault interrupt to the peripheral processor, which branches to the relevant routine.

2. Communication Mode: Specialized communications interface hardware is provided in both machines. The two machines use their respective communication instructions to converse and to interrogate each other and to determine each other's requirements. The vector machine's memory control unit is provided with a vector processor-peripheral processor communication register, J_{XW}, 64 bits long. The vector processor's operation control unit is provided with a peripheral processor-vector processor communication register, J_{WX}, 64 bits long. The communication control signals transmitted between the two machines include the relevant communication interrupt request signals and the corresponding response signals. Both machines use their respective communication instructions to initiate communications interfacing.

The vector processor's system instructions include two communication instructions used exclusively by the core state: an instruction by which the peripheral processor sends information to the vector processor, WCJ → XLJ, and an instruction by which the vector processor sends information to the peripheral processor, XLJ → WCJ. When the instruction WCJ → XLJ is executed, after the contents of the outgoing communication register J_{WX} are received by the vector processor, it sends a response signal to the peripheral

processor and clears the "J_{WX} busy" flip-flop to zero. When the instruction XLJ → WCJ is executed, after the vector processor sends information to the outgoing communications register J_{WX}, its busy flag is set at 1; after the information reaches J_{WX}, it sends a communications interrupt request signal to the peripheral processor.

The peripheral processor's instruction set is similarly provided with two special core state communication instructions: the incoming communication instruction #TR and the outgoing communication instruction #TC. When #TR is being executed, after the peripheral processor has received the contents of the outgoing communications register J_{XW}, it sends a response signal to the vector processor and clears the "J_{XW} busy" flip-flop to 0. In executing #TC, the peripheral processor sends the information to the communications interface J_{WX} and sets its busy flat at 1; after the signal is sent to J_{WX}, it sends a communications interrupt request to the vector processor.

The two processors thus use communications instructions and the interrupt mode for communication and response.

3. The Batch Transmission Mode: The vector machine and peripheral processor each have a main storage; the main storages are not shared. In order to allow rapid batch information exchange between the two main storages, a direct data channel is provided between the two machines. For the peripheral processor, this direct data channel is 1 of its 32 subchannels (see Figure 1).

The vector machine's memory control unit is provided with seven main storage access channels: correspondingly, it has the disk channel 0 request P_0Q, the disk channel 1 request P_1Q, the instruction control unit instruction fetch channel request ZQ, the look-behind storage data wait station channel request HQ, the memory control unit instruction stack buffer register channel request ZDQ, the peripheral processor 0 channel request W_0Q, and the peripheral processor 1 (expandable) channel request W_1Q (see Figure 2). These seven channels are time-share queued into the instruction control unit pipeline stations in accordance with a priority system. Because the peripheral processor is slower than both the vector processor and the disk units, and signal exchange is not timed, so that the peripheral processor has the lowest priority level in the time-shared queue.

The specialized hardware interfaces provided in the vector processor memory control unit for the direct data channel to the peripheral processor include a data buffer station and a control command word register. The data buffer station consists of 16 64-bit shift buffer registers WH (Figure 3). Its inputs can come from the peripheral controller code line or from the memory control unit's internal storage data readout buffer J_{ds}. Its output can be sent via long lines to the peripheral controller or can be sent to the memory control unit's internal storage data write register J_{XS}. The data codes in WH can be transferred in or out by the peripheral device controller's synchronization pulses, or by the memory control unit's condition pulses. The control word register includes the request address register WQD, the local exchange number registers WL, and the read-write flag WXM.

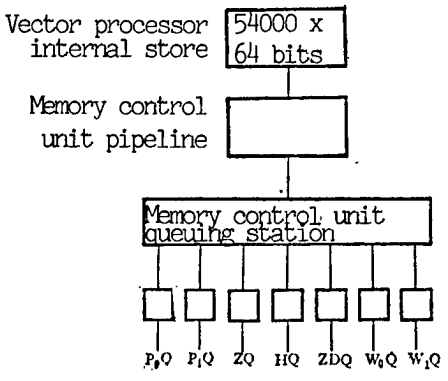


Figure 2. Internal Access Channels of the Vector Processor Memory Control Unit

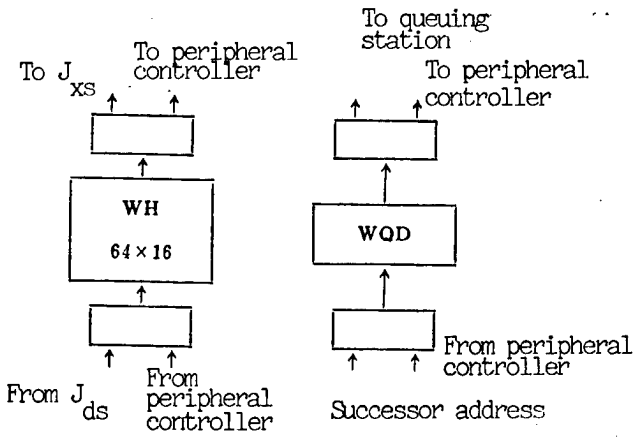


Figure 3. Vector Processor's High-Speed Channel Interfaces With Peripheral Processor

Reading from the vector machine's internal storage and transmitting the result to the peripheral processor is performed as follows. After the peripheral controller has sent the starting address of the data exchange with the peripheral processor, the exchange length, and the read-write flag, it sends the preliminary request signal Z"1"C_{WYQ} to the memory control unit. It then waits until the vector processor memory control unit has completed the exchange and has sent the response signal C_{WHD}, and uses this response signal to control the sending of the next request signal. After the memory control unit receives the preliminary request signal from the peripheral controller, it loads this signal into the peripheral processor request flip-flop C_{WQ}, synchronized by peripheral processor pulses. It then waits while the memory control unit performs time-shared queuing. After the current request is put into the queue, the memory control unit produces a load instruction NC → W which enters the various stations of the memory control unit's pipeline and successively reads out a quantity WL of data words from internal storage and loads them into WH. After this operation is completed, the memory control unit sends an "exchange completed" response signal to the peripheral controller. Then, again in accordance with the peripheral controller synchronization pulses, it successively transfers the data in WH to the peripheral controller's receiving register. After WL components have been exchanged, the memory control unit forms the successor address and loads it into WQD. If the request which the peripheral controller sends on the next occasion does not include a new request address, the successor address begins to be exchanged from WQD until one data area has been completely exchanged.

Writing into the vector processor's internal storage by the peripheral processor: After the peripheral controller transfers the request control word to the memory control unit, the exchange codes of the WL numbers which are to be written into the vector processor's internal storage are transferred into WH in accordance with the controller synchronization pulses. Then the peripheral controller sends a preliminary request signal Z"1"C_{WYQ} to the memory control unit. After the memory control unit receives this signal

from the peripheral controller, in synchrony with the memory control unit's pulses, it stores the preliminary request signal in the peripheral processor's channel request flip-flop CWQ and places it in a time-shared queue. Once the current request has been placed in the queue, the memory control unit emits the stored instruction W → NC which enters the stations of the memory control unit's pipeline; the W → NC instruction successively writes a number WL of data words which are in WH into WL memory cells in main storage, beginning with WQD. After the writing is completed, the memory control unit sends an "exchange completed" response signal C_{WHD} to the peripheral controller. The peripheral controller uses the response sent by the memory control unit to control preparations for the next exchange request. similarly, provided that the peripheral controller is continuously sending request signals, the successor address which the memory control unit saves in WQD can carry out exchange of a data area.

III. Information Exchange Between the Vector Processor and the Disk-Tape Channels

In order to increase the data transmission rate and decrease demands on the peripheral processor, the vector processor and peripheral processor share disk and tape resources. The system also has two high-speed direct data channels between the vector processor and the magnetic disk units, magnetic tape units, electrostatic printer, and graphic display unit. The eight subchannels Disk 0, Tape 0, Tape 2, Electrostatic Printer, Disk 1, Tape 1, Tape 3, and Graphic Display are divided into two groups; the first four are combined into one path which forms a direct channel for information exchange with the vector processor's main memory and is called the Disk 0 channel for brevity; the latter four subchannels are also formed into a path, comprising the second direct data channel, which is called Disk 1 for brevity. To meet the requirement for direct access to the vector processor, and in particular to adapt to the speed of the disk unit, a special controller is provided; this is generally called the disk-tape channel controller or, for brevity, the disk controller.

Because the disk-tape channels are clocked, they have the highest priority level within the vector processor memory control unit among the seven internal storage access request channels. Similarly, the disk-tape channel operates under the control of the peripheral processor and the vector processor is passive. If the operating frequency of the disk unit is 8 MHz, 215 ns is required to read out each bit of information from the disk. Since one word is 64 bits, 16 μs is required to read out a word. Thus, it takes 256 μs to exchange 16 words of data. The maximum time elapsing from the vector processor memory control unit's reception of the request signal sent out by the disk channel to the end of transmission of 16 words of data is 73 μs (with a vector processor operating frequency of 10 MHz). Under these circumstances, to accelerate the transmission rate, two data buffer shift registers have been provided for each disk channel at the vector processor's memory control unit interface; each has a capacity of 64 bits x 16 words, and they are called Disk Buffer 0 and Disk Buffer 1 (PH_0 and PH_1), as shown in Figure 4. The two disk buffers can receive data read out of internal storage or data from the disk controller; both of them can also send data

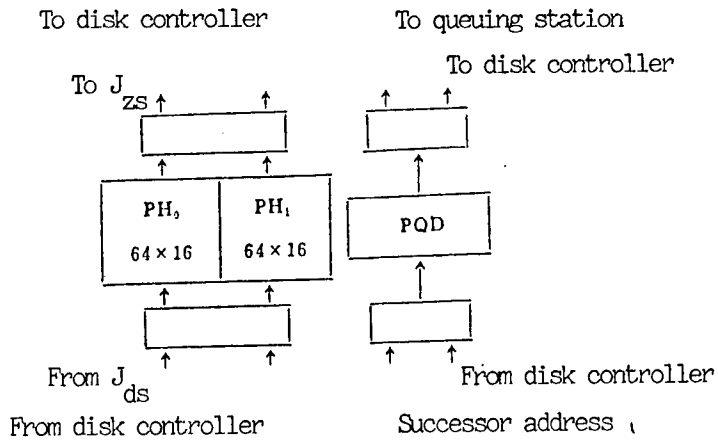


Figure 4. High-Speed Channel Interface Between Vector Processor Memory Control Unit and High-Speed Disk-Tape Channel

to internal storage or the disk controller. In addition they are capable of parallel alternate operation. For example, when reading from the vector processor's internal storage and writing on the disk, while the 16 words in one disk buffer are being written onto the disk, the other disk buffer can be receiving the next batch of 16 words from internal storage in preparation for writing onto the disk. When reading from the disk and writing into the vector processor's internal storage, while the 16 words of data in one disk buffer are being written into internal storage, the other disk buffer can be receiving the next batch of 16 words from the disk in preparation for writing into internal storage.

As in the case of the direct data channel to the peripheral processor, the memory control unit has been provided with a request control word register for the disk channels, which includes the disk request address register PQD, the exchange length register PL, and the read-write flag PXM.

Reading main storage and writing onto disk, NC → P: After the disk controller sends the request control information, i.e., the starting address, exchange number, and read-write flag, for exchange with the vector processor internal memory, it sends the preliminary request signal Z"1"C_{PYQ} to the memory control unit. After the memory control unit receives the preliminary request signal, the signal is sent to the disk channel request flip-flop C_{PQ} in synchrony with the memory control unit's pulses, where it is placed in the memory control unit access channels queue. After it is placed in the queue, the memory control unit produces a load instruction NC→P which enters the various stations of the memory control unit's pipeline and reads out 16 bits of data from the internal storage (or a number PL of words) and sends them to the disk buffer. Then the memory control unit sends out another request and reads from internal storage the adjoining 16 words of data, which are sent to the other disk buffer. When the two disk buffers are full, the memory control unit sends the response signal $C_{P_{HD}}$ to the memory control unit. When

the disk controller receives the response signal, its synchronization pulses are used to transfer the data in one disk buffer to the disk controller's receiving registers; after all of the data is transferred out of one disk buffer, another request to read from internal storage onto the disk is sent to the memory control unit. After the memory control unit receives the request, it reads out 16 words of data from internal storage and forwards them to the disk buffer that has just been emptied, as outlined above. At the same time, the disk controller is transferring the 16 words from the other disk buffer to the disk controller to be written onto the disk. In this parallel alternate operation by the two disk buffers, emission of the successor address is controlled by the memory controller until the data in a data area within internal storage have been entirely read out onto the disk.

Reading from disk into the vector processor internal memory, P → NC: After the disk controller sends a request control word signal to the memory control unit, the disk controller's synchronizing pulses are used to shift the 16 words of data (or PL words of data) read out from the disk into the disk buffer, after which a preliminary request signal Z"1"C_{PYQ} is sent to the memory control unit. After the memory control unit receives the request signal, the memory control unit's pulses synchronize the loading of the request signal into the disk channel request flip-flop C_{PQ} where it is placed in the memory control unit's internal storage access channel queue. After it is placed in the queue, the memory control unit it produces a write data instruction P → NC, which enters the stations of the memory control unit's pipeline and reads the 16 words of data in the disk control unit successively into memory; when this is completed, it sends a response signal C_{PHD} to the disk controller. When the memory control unit is writing the 16 words from one disk buffer into internal storage, it is also transferring the data read out from the disk into the other disk buffer. When the disk controller receives the response signal from the memory control unit and the data in the other disk buffer are ready, it sends a write into main storage request to the memory control unit. After this request is sent out, one disk buffer writes the data into main memory, while the other disk buffer, which has already been emptied, under the control of the disk controller, prepares the next batch of data for writing into main memory. This parallel alternate operation of the two disk buffers is well suited for increasing disk speed and effectively increases data transmission rates.

In order to control the parallel in-and-out alternations of the two disk buffers, the memory control unit is provided with two corresponding control flip-flops which are called the disk, buffer in flip-flop (C_{PHR}) and the disk buffer out flip-flop (C_{PHC}) (see Figure 5). They are both D-type flip-flops. C_{PHR} = 1 puts PH_1 into the entry state; C_{PHR} = 0 puts PH_0 into the entry state. C_{PHC} = 1 puts PH_1 into the output stage; C_{PHC} = 0 puts PH_0 into the output state. When the system is started up (and the entire machine is set at 0; i.e., QJZO), C_{PHR} = 0 and P_{PHC} = 1. Thereafter, every time an enter pulse arrives, the two flip-flops change state and control the input and output gates of the two disk buffers so that they are alternatively in the input and output states.

When the disk buffer is writing onto the disk, the control voltages emitted by the two disk buffers are:

$$PH_0SP = \overline{PDHS} \wedge \overline{C_{PHC}}$$
$$PH_1SP = \overline{PDHS} \wedge C_{PHC}$$

When the disk buffers are writing into internal storage, the control voltages emitted by the two disk buffers are:

$$PH_0XS = \overline{PDHS} \wedge \overline{C_{PHC}}$$
$$PH_1XS = \overline{PDHS} \wedge C_{PHC}$$

When the disk is writing into the disk buffers, the control signals sent to the two disk buffers are:

$$PXPH_0 = PXM \wedge \overline{C_{PHR}}$$
$$PXPH_1 = PXM \wedge C_{PHR}$$

When internal memory is writing into the disk buffers, the control voltages sent into the two disk buffers are:

$$NXPH_0 = (NCSP)6 \wedge \overline{C_{PHR}}$$
$$NXPH_1 = (NCSP)6 \wedge C_{PHR}$$

Here PDHS is the address recovery voltage sent by the disk controller. It is used to control the selection of whether the information sent via the code line to the disk controller is the main machine internal storage request (PQD) or the code to be exchanged (PH_0 or PH_1). (NCSP)6 indicates that the sixth station in the memory control unit's pipeline holds an instruction for internal storage to transmit to disk; at this time the data which is read from internal memory into the memory control unit's data read register J_{ds} should be forwarded to the disk buffer.

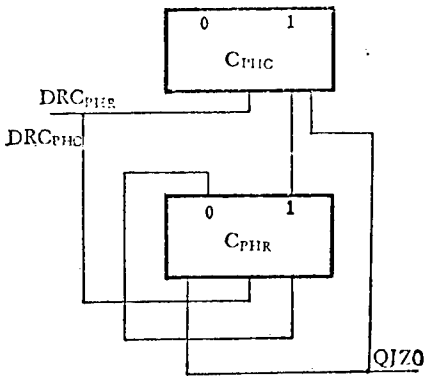


Figure 5. Control of Two Disk Buffered Input/Output Parallel Alternation

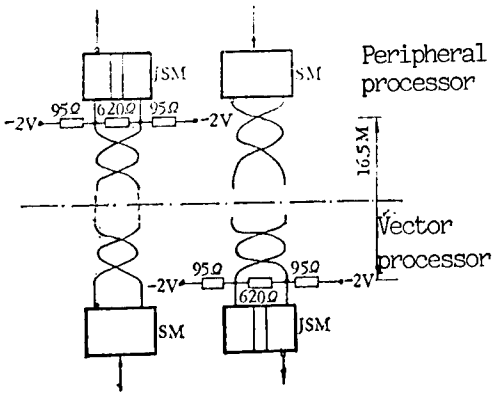


Figure 6. Long-line Transmission of the Vector Machine and the Peripheral Processor

IV. Implementation of Information Exchange by Long-Line Transmission

All of the logic circuits in the 757 vector processor use Chinese-made high-speed ECL [emitter-coupled logic], medium- and small-scale integration circuitry, but the peripheral processor's logical circuitry uses TTL [transistor-transistor logic] integrated circuits. Therefore, the transmission of data and control signals between the vector processor and peripheral processor must undergo E-T or T-E conversion. In the current system, the vector processor is installed downstairs, and the peripheral processor and peripheral devices upstairs, with information transmission lines up to 16.5 meters long. In order to assure reliable transmission, the long-line transmission makes use of twisted-pair duplex transmission of ECL signals (Figure 6). The sending end uses a dual gate SM as a duplex send gate. The receiving end uses an ECL receiving gate JSM to receive the dual-gate ECL signals from the long lines. All E-T and T-E voltage level conversion is carried out by the peripheral processor. In order to match the resistance of the entire ECL system, three-resistor matching is used in the receiving gates at the end of the long lines.

In order to save layout space these three resistors are also realized as integrated circuit resistor networks. In addition, long-line twisted pair transmission lines of suitable characteristics were also specially designed. Theoretical analysis, computer calculations and experience have shown that this type of transmission and matching method gives a small reflection factor, little signal crosstalk, little waveform distortion, and low voltage losses. Prolonged reliable operation of the entire machine has also confirmed that the design of the transmission system is entirely reliable.

The method by which data and control information are transmitted between the vector processor and the disk-tape channel is entirely analagous to the method of transmission between the vector processor and the peripheral processor described above and will not be discussed further.

8480/9365
CSO: 4008/200

AUTOMATED HARDWARE FAULT PROCESSING ROUTINES OF 757 COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 54-57

[Article by Zhang Shuwen [1728 3219 2429], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] I. Introduction

Malfunctions in computer operation may cause major losses. Accordingly, when designing a computer system, it is important to give it protective redundancy and to enable the system to go on operating correctly or at least not to become paralyzed when hardware or software faults occur. Fault tolerance was taken as an objective in the system design process, and hardware and software were integrated in order to implement automatic handling of hardware faults, so that when malfunctions occurred the opportunity to continue running the current job was maximized, or if this was impossible, the likelihood of system paralysis was minimized. As a result of this approach, most nonpermanent faults can be overcome and will not affect normal job processing. In the case of permanent faults which cannot be dealt with, the malfunction can be located in timely fashion, thus greatly decreasing machine maintenance time and extending the time between failures, which saves time and manpower and increases machine availability. These factors are particularly important for large high-speed computers. In general,

$$A = \frac{MTBF}{MTBF + MTTR}$$

where A is the availability (usability), percent; MTBF is the mean time between failures; and MTTR is the mean time to repair.

The smaller the number of malfunctions and the greater the probability of their recomputability, and in addition the shorter the repair time, the greater the efficiency of the machine will be.

In addition, we made some fundamental alterations in fault management. For many years the maintenance personnel were expected to maintain a log, keep malfunction statistics, analyze faults, and identify and repair them. Not only was this laborious, but the correctness and detail of the records varied from individual to individual. In order to overcome these

deficiencies in the 757 described here, in addition to implementing automatic fault processing we also instituted automatic detailed logging of faults. Such information as the time of occurrence and location of the fault, its nature, the method of dealing with it and the results achieved are all recorded in detail. In this way, the maintenance personnel can make regular statistical analyses and can receive information on faults at various stages simply by using the keyboard or typewriter. It is also possible to keep statistics automatically, to analyze the characteristics and causes of faults that occur, and to increase the quality of machine maintenance and management; these factors also helped improve design.

II. Fundamentals of Hardware Interrupt Processing in the 757 Mainframe

In the 757 the peripheral processor (WCJ) diagnoses the mainframe (XLJ). The mainframe's assignments and states are recovered and subsequently cleared by the "communication out" (TC) instruction and the "communication in" (TR) instruction in the peripheral processor. A small fan-in-fan-out unit (SRC) is installed between the peripheral processor and mainframe. The TC instruction gives the mainframe and the fan-in-fan-out unit (SRC) a series of instructions and data which are analyzed by the SRC: if they are instructions, then a series of control voltages is produced, which control the mainframe's fan-in or the information saved from control gates, while if they are data, then they are fanned into the mainframe's corresponding registers or flip-flops. The TR instruction causes the states recovered from the mainframe to be stored in the peripheral processor's internal memory.

Parity check points are provided at certain locations in the various units of the mainframe, and Hamming code test points at some key locations. Clear, fan-in and voltage save are provided at key flip-flops and registers in order to allow assignment and sensitization of certain paths. When a hardware fault is discovered, the appropriate control unit's hardware fault interrupt message is sent out, the clock in the control unit where the fault has occurred immediately stops, and all pulses are suspended throughout the machine. An interface is provided between the malfunctioning control unit and the other control units so that information coming from the other control units is not lost. This makes it possible to save the current state. In addition, certain save stations are provided so that after the stoppage occurs it is possible to reconstruct useful states. After a hardware fault is discovered in the vector processor, a "mainframe fatal fault" interrupt is sent to the vector processor.

A diagnostic unit switch is provided on the mainframe's maintenance and control panel, together with retry switches for the various control units. When it is not desired to diagnose operation, the diagnosis unit can be cut out and hardware faults may be processed without diagnosis. If for various reasons to double calculate is impossible in one of the control units, the retry switch for that unit can be used to cut out double calculation, while double calculation can still be carried out as usual in the other control units.

III. Control Connections Between the Operating System and the Diagnostic Interrupt Programs

When the peripheral processor WCJ receives a mainframe fatal fault interrupt, the operating system informs the diagnostic interrupt program of the status of the peripheral processor's internal storage (Figure 1) and transfers control to the main diagnostic interrupt control program.

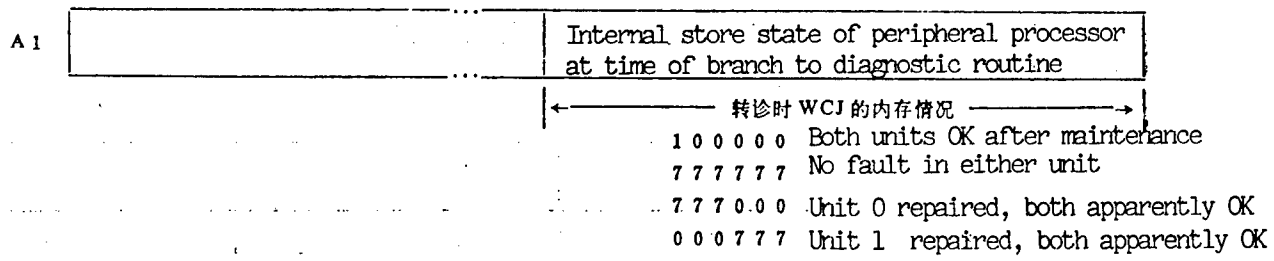
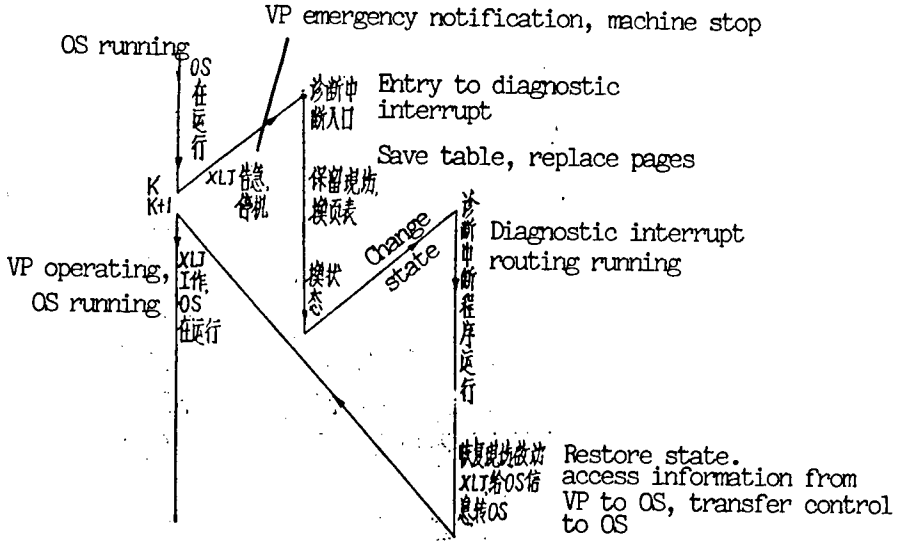


Figure 1.

The work of the main diagnostic interrupt control program starts with saving of the states of the peripheral processor and mainframe in order to assure their correctness and guarantee that the job currently being run can be resumed; if this is impossible, it tries to prevent system paralysis. Human intervention is minimized, and an effort is made to do everything automatically, which increases machine efficiency.

The control switching between diagnosis and the operating system is diagrammed in Figure 2.



Note: VP: Vector processor
OS: Operating system

Figure 2.

The operating system switches from the core state to the diagnostic interrupt program. After branching to this program, it saves the old status word and the old interrupt entry as well as the contents of the communications interrupt flag bit C_{DXW} and the communications register (WX). It changes to a new interrupt entry point, changes the page table, and changes to the new status word, after which the diagnostic interrupt program operates in the diagnostic mode, and the peripheral processor begins to carry out the various types of TC instruction processing in which each time it carries out a TC instruction it initiates operation of the SRC [fan-in-fan-out unit] and sends the contents of WH [communication register] to the SRC unit. Under control of the SRC unit, it performs assignment, recovery and clearing in the mainframe. On performance of the diagnostic interrupt, the diagnostic interrupt lamp lights.

IV. Fault Processing

After the diagnostic interrupt program has saved the current state, by analyzing the mainframe interrupt word and the fault detection word of the relevant control unit it can find the reason for mainframe stoppages, and accordingly it has a variety of methods of handling them. When the error state can be reconstructed and accordingly is recomputable, it branches to the retry program of the control unit in which the fault occurred, and after the retry is performed it returns to the diagnostic interrupt program and starts the mainframe; at this point it is still under the control of the diagnostic interrupt program. After a successful double-calculation, the mainframe is started and waits for logging to be performed; after the machine stoppage condition is cleared the mainframe status is restored, and if the entire machine is fault-free the mainframe is started and control is transferred to the operating system.

If the double-calculation is not successful, another is attempted. During each double-calculation, in addition to execution of the program itself there is also a delay of 200 msecs; it is assumed that after 7 double-calculations an intermittent fault should have disappeared. After a successful double-calculation, control is transferred to the operating system. If another malfunction occurs, the transfer to the operating system is made after further processing. If the malfunction that occurs is nondouble-calculable, then alternate programming is undertaken (switch to another job). If the same fault appears repeatedly, a branch is made to the diagnostic program of the unit in which the fault occurs in order to locate it, and the number of the card that is identified, the number of the control unit involved, and the time at which the malfunction occurred are shown on the display; after the maintenance technician has replaced the card, he uses the keyboard to type an order and to report the number of the card that has been replaced (the malfunction has also been automatically logged) and the relevant diagnostic program verifies the replacement. Once the fault is eliminated, the system is returned to alternate processing. When the mainframe is started, it is in the general stop [guang ting [1639 0255]] state, waiting for control signals to be sent from the operating system in the peripheral processor. The return is made to alternate processing on the basis of information from the diagnostic routine.

V. Exit From the Diagnostic Interrupt Routine

After the diagnostic interrupt program has carried out double calculations or diagnosis, it sends information to the operating system informing it of the status of the mainframe's internal storage, the status of exchange with peripherals, the retry status, the state of the mainframe at the time of the stoppage and the like, so that the operating system can take the proper steps; this information is sent via accumulator A (see Figure 3).

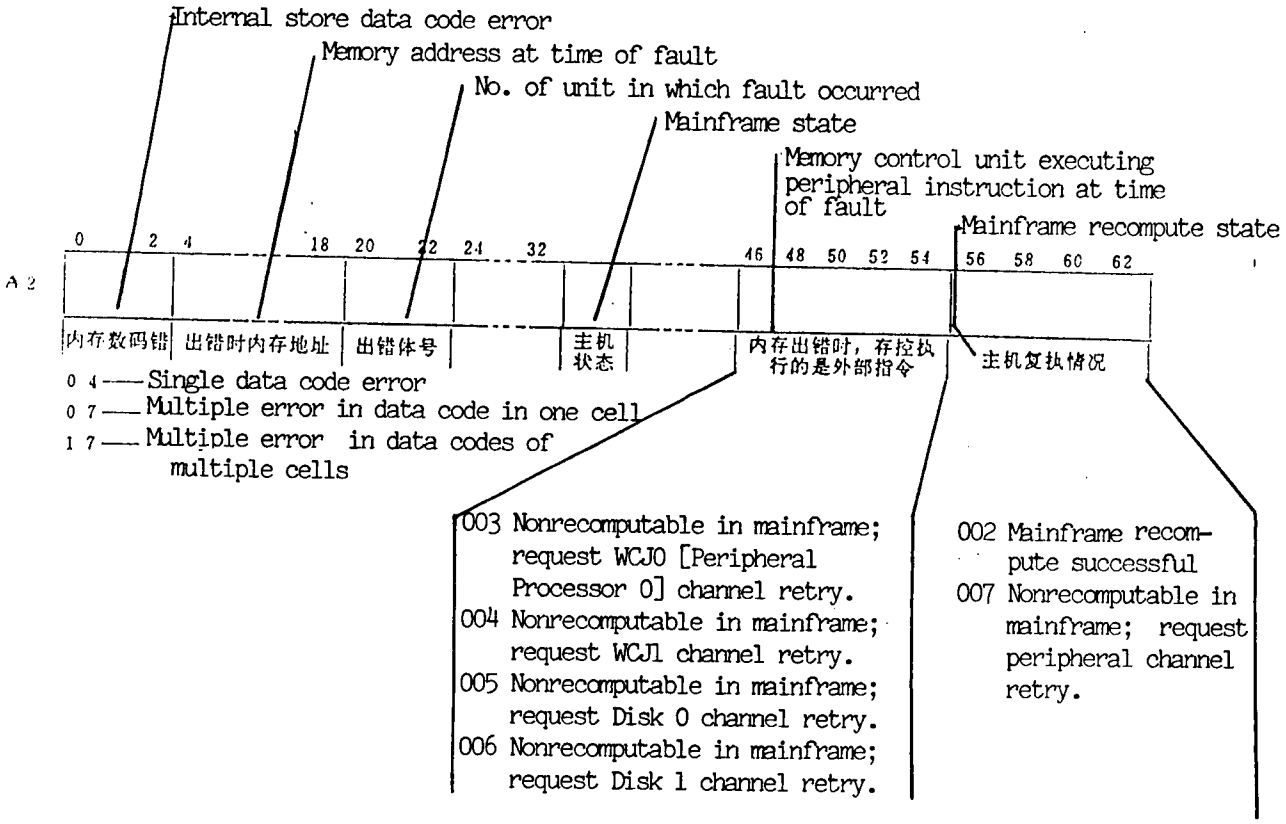


Figure 3. Information Sent by the Diagnostic Interrupt Program to the Operating System

Because the WCJC (peripheral processor) and the XLJ (mainframe) operate asynchronously with respect to each other, restoration of state and starting of the mainframe must be processed in the proper sequence; otherwise errors will result and the system will not operate normally. In addition, the number of jumps, between the operating system and the diagnostic interrupt program must be minimized. After branching to the diagnostic interrupt, if possible when processing is completed the mainframe should be turned over to the operating system in fault-free condition; this makes for correctness and can increase efficiency.

[Following two sentences garbled in original.] Communications interface register WX (in the peripheral processor) and the saving and restoration of communications interrupts. In the nondiagnosis state, register WX functions as a communications interface register between the peripheral processor and the mainframe, and during communications when the mainframe produces a communications interrupt (bit 29 in interrupt word T 1[29]). When the mainframe stops, it may be at various stages in the processing of a communications interrupt or may not yet have begun processing. In the diagnostic state, WX sends information from the peripheral processor to the SRC [fan-in-fan-out unit and the mainframe, and accordingly the problem of releasing and recovering communications interface WH arises; continuity in communications interrupt processing must be maintained in order for the system to operate correctly.

VI. Emergency Notification by Operator

If a diagnostic program must be executed during system operation, a fault record is displayed (or printed out); when a fault record is displayed or one unit is substituted for another, it is necessary to enter the diagnostic state. The "emergency notification by operator" mode can be used to run the diagnostic interrupt program, after which a keyboard command is used to enter a request.

A notification of emergency can be made by pressing the "emergency notification" button on the mainframe maintenance panel or the peripheral processor monitoring and control panel. When there is no malfunction in the mainframe, after the keyboard command is executed, it can continue operation without disrupting execution of the job, provided that the state of the mainframe has not been disturbed.

VII. Fault Logging and Display

Each time a malfunction develops in the mainframe or it stops because of an emergency notification by the operator, the time of the malfunction and its detailed location must be recorded.

The memory contains two pages for diagnosis; 256 units are allocated for fault logging, and if these 256 units are filled the information is stored on disk as a record. The date of the first fault is used as the name of the record. and when necessary it can be read out and printed or displayed. In addition, fault records showing the details of card replacement can be printed out.

The individual items in a fault record are shown in Figure 4.

The principle on which the design of this diagram is based is that of dealing with the machine stoppages caused by hardware faults in the mainframe as clearly as possible. The time of emergency notification by the operator is also logged. In addition, the number of faults in a given control unit in the mainframe can be displayed at signal light T_3 on the control panel, making it easy to determine the fault frequency. These features give a clear indication of fault circumstances in the mainframe, increase the quality of maintenance, and avoid time-consuming work for the maintenance personnel.

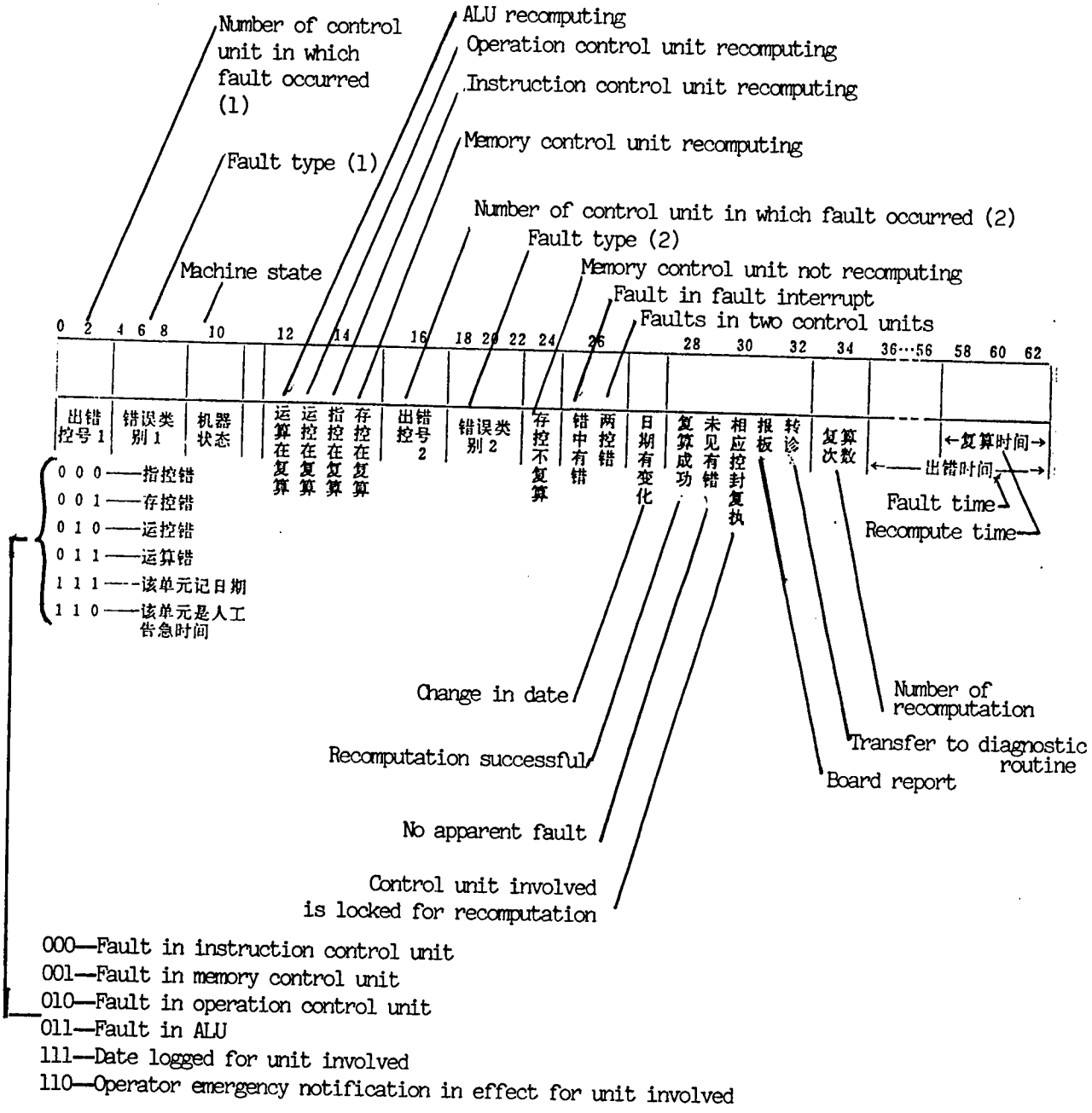


Figure 4. Fault Logging Record

VIII. Conclusions

Fault-tolerant design with software-hardware integration is implemented in this machine. We have implemented rather good automatic processing of hardware faults, with double-calculation or diagnosis and fault location, and have provided automatic fault management and logging, which has decreased operator intervention and improved efficiency. This is extremely helpful in increasing machine reliability, maintainability, and efficiency. Other

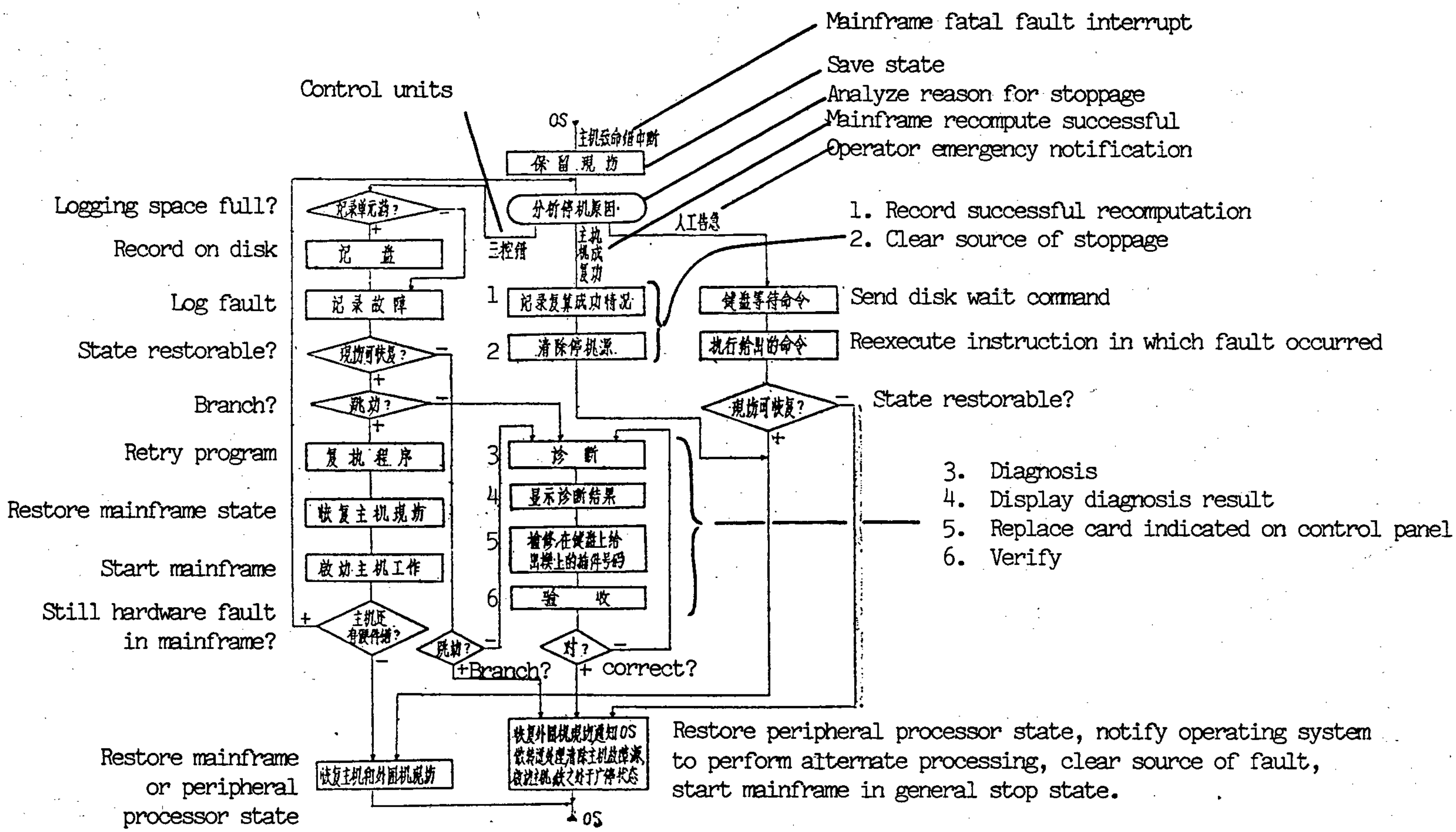

Control units
Mainframe fatal fault interrupt
Save state
Analyze reason for stoppage
Mainframe recompute successful
Operator emergency notification
OS
Logging space full?
Record on disk
Log fault
State restorable?
Branch?
Retry program
Restore mainframe state
Start mainframe
Still hardware fault in mainframe?
1. Record successful recomputation
2. Clear source of stoppage
Send disk wait command
Reexecute instruction in which fault occurred
State restorable?
3. Diagnosis
4. Display diagnosis result
5. Replace card indicated on control panel
6. Verify
Branch?
correct?
Restore mainframe or peripheral processor state
Restore peripheral processor state, notify operating system to perform alternate processing, clear source of fault, start mainframe in general stop state.
OS

Figure 5.

effects are sure to manifest themselves during future machine operation. This obviously constitutes gratifying progress in fault-tolerant design, but it is still far below the level of the fault tolerant design of some machines in the world arena, and there are many points that still merit discussion and improvement.

8480/9365
CSO: 4008/200

DESIGN OF MULTIBIT COMPARATOR FOR 757 COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 2, 1984 pp 58-60

[Article by Liu Yulin [0491 3768 2651], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] Abstract. Comparators are extremely widely used in digital computer logic circuitry and digital devices. Multibit comparators can be made by cascading 2-bit and 1-bit comparator elements, but they are rather slow. Another possibility is independent design, but the problem is that there are too many variables and it is rather toilsome to simplify the functions involved. This article describes a 19-bit comparator designed for the 757 large-scale computer's instruction control unit, as well as the design method that was used.

I. Design Requirements

The instruction control unit of the 757 computer contains three counters to dispatch and fetch instructions; these three counters are the lower boundary address counter X, the instruction address counter JS_Z, and the upper boundary address counter S. When an instruction is dispatched from internal storage to the instruction buffer, the upper boundary address counter S points to the internal storage address, and the lower boundary address counter X and the upper boundary address counter S are used to define its location. When an instruction is fetched from the instruction buffer storage to the instruction register, the instruction address counter JS_Z points to the instruction buffer memory address. In many branch instructions, if the branch address Q formed by the computer during operation is equal to or greater than the upper boundary address S (the contents of the upper boundary address are also denoted by S) or if it is smaller than the lower boundary address X (the contents of the lower boundary address are also denoted X), then a fault immediately occurs, and this requires location by the address counters X and S again, after which the instruction is again fetched. These circumstances, i.e., when $Q \geqslant S$ or $Q < X$, are called "branching out of area." A circuit which has the capability to determine whether $Q \geqslant S$ or $Q < X$ is a 2-bit comparator. If the comparator itself produces an error, this may cause

chaos in the instruction control unit. Thus, it is obvious that the operation of the comparator is extremely important: it must be not only stable and reliable, but fast.

Computers which we designed previously made very little use of multibit comparators; considerable use was made only of coincidence circuits, which could compare two multibit binary numbers (addresses) and determine whether they were equal. The gate combinations of coincidence circuits are very easy to implement.

There is little information on the design of digital comparators in ordinary books on digital computer principles; it is described in books on the logical design of digital systems. When it is desired to compare two binary numbers, the guiding idea is always bit-by-bit comparison; in some designs the comparison proceeds from the highest to the lowest bit, in other designs from the lowest to the highest. Most multibit comparators use cascaded 2-bit or 1-bit comparators, and accordingly are very slow, so that they do not meet design requirements. The same is true of comparators of modular design. Another possibility is design of independent [i.e. not built up of smaller units] multibit comparators, but the problem is that there are too many variables and it is toilsome to simplify the functions. For example, a 4-bit comparator involves an 8-variable logic problem, and it is difficult to use Karnaugh maps to represent it. Thus, the difficulties involved with a 19-bit comparator can readily be imagined. In addition, multilevel circuitry can be used to implement certain functions, or the factoring method can be used, but the large equations may have many factors, the circuits to implement them have completely different characteristics and it is very difficult to derive the formulas involved.

To summarize the above, we analyzed the branch-out-of-area conditions of the 757 large-scale machine's instruction control unit and used the following approach to implement a multibit comparator based on binary computations.

II. Design of the Multibit Comparator

The basic logic circuits used in the comparator were the ECL-D system of medium- and small-scale integrated circuits produced by the Chinese Academy of Sciences' Plant No 109. They included single gate, quad gate, and quad half-adder circuits.

Let Q, X, and S represent 3 19-bit binary numbers:

$$Q_0Q_1Q_2\cdots Q_{16}Q_{17}Q_{18}$$

$$X_0X_1X_2\cdots X_{16}X_{17}X_{18}$$

$$S_0S_1S_2\cdots S_{16}S_{17}S_{18}$$

It is required to design 2 19-bit comparators for the functions $Q < X$ and $Q \geqslant S$. This can be realized by means of the following type of implementation.

Comparator Operating Principles

If Q and X are to be compared, we perform the subtraction Q - X and carry out the fixed point complement addition $Q + [-X]_{comp} = Q + [-X]_{radix-1\ complement} + 1$, ignoring the computation result and taking account only of whether the adder produces a carry; then if $Q < X$ the adder will not produce a carry and the C_j output bit will be at the low voltage level (-1.6 V). Thus the $\overline{Q < X}$ output will be low and the $Q < X$ output will be high (-0.8 V), indicating that the condition $Q < X$ exists (see figure). Otherwise, the adder will produce a carry and the carry output C_j will be high, so that the $\overline{Q < X}$ output will be high and the $Q < X$ output will be low, indicating that the condition $Q < X$ does not exist.

If Q and S are to be compared, we perform the subtraction Q - S and perform fixed point complementary addition $Q + [-S]_{comp} = Q + [-S]_{radix-1\ complement} + 1$, again ignoring the addition result and noting only whether the adder produces a carry. If $Q \geqslant S$, then the adder produces a carry and the carry output bit C_j will be high; accordingly the $Q \geqslant S$ output (equivalent to $\overline{Q < S}$) will be high and the $\overline{Q \geqslant S}$ output (equivalent to $Q < S$) will be low, indicating that the condition $Q \geqslant S$ exists. Otherwise, the adder will not produce a carry, and the carry output bit C_j will be low, so that the $Q \geqslant S$ output will be low and the $\overline{Q \geqslant S}$ output high, this means that the condition $Q \geqslant S$ does not exist.

In the multibit comparator we used only the relevant parts of the adder carry chain, which saved hardware and increased reliability. The speed of the comparator depends on the type of carry chain. We need to select a carry chain type suited to the comparator characteristics, and in general increasing the speed will require an increased amount of hardware.

We now describe the derivation of the logic design equations: Let C_j, C_0, C_1, C_2, $\cdots$, C_{15}, C_{16}, C_{17} represent the output carry bits resulting from comparison of 2 19-bit binary numbers, and let C_{18} represent the carry bit C_{+1} from the low-order bit ($\bar{C}_{+1}$ = -1.6 V).

If half adders are grouped into a full adder, the half-sum equation is

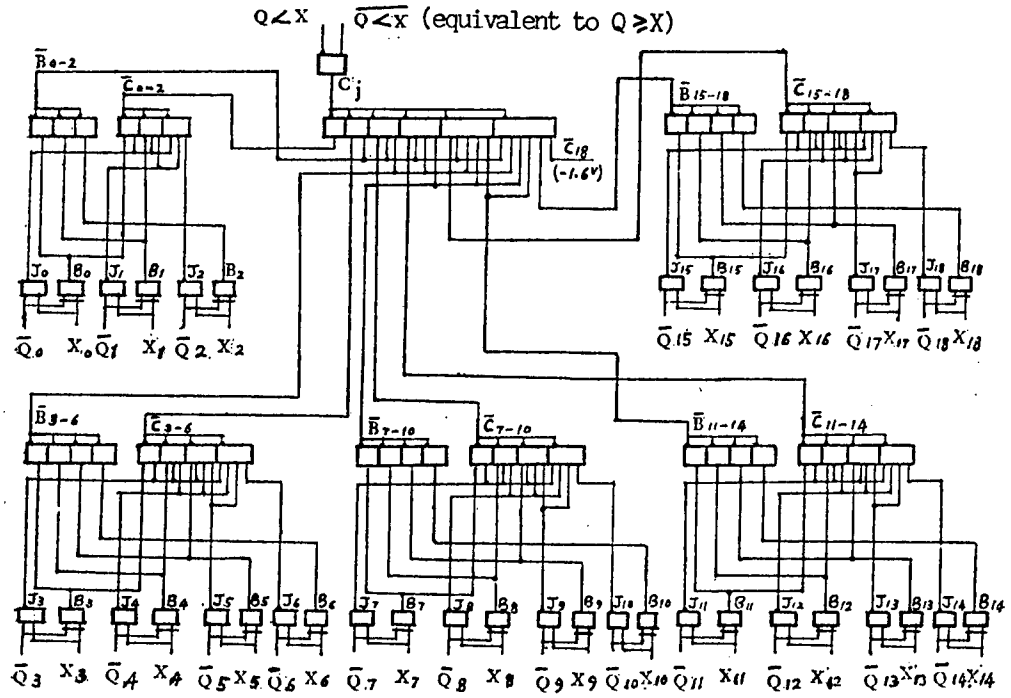
$$B_i = Q_i \oplus X_i \tag{1}$$

the half adder carry equation is

$$J_i = Q_i X_i \tag{2}$$

and the full adder carry equation is

$$C_{i-1} = J_i + B_i C_i \tag{3}$$



Circuitry of Multibit Comparator

From equation (3) and the laws of Boolean algebra we can derive the equations

$$C_{17}=J_{18}+B_{18}C_{18}$$

$$C_{16}=J_{17}+B_{17}C_{17}=J_{17}+B_{17}(J_{18}+B_{18}C_{18})=J_{17}+B_{17}J_{18}+B_{17}B_{18}C_{18}=(J_{17}+B_{17})(J_{17}+J_{18})+B_{17}B_{18}C_{18}$$

$$C_{15}=J_{16}+B_{16}C_{16}=J_{16}+B_{16}(J_{17}+B_{17})(J_{17}+J_{18})+B_{16}B_{17}B_{18}C_{18}=(J_{16}+B_{16})[J_{16}+(J_{17}+B_{17})(J_{17}+J_{18})]+B_{16}B_{17}B_{18}C_{18}$$
$$=(J_{16}+B_{16})(J_{16}+J_{17}+B_{17})(J_{16}+J_{17}+J_{18})+B_{16}B_{17}B_{18}C_{18}$$

$$C_{14}=J_{15}+B_{15}C_{15}$$
$$=J_{15}+B_{15}(J_{16}+B_{16})(J_{16}+J_{17}+B_{17})(J_{16}+J_{17}+J_{18})+B_{15}B_{16}B_{17}B_{18}C_{18}$$
$$=(J_{15}+B_{15})[J_{16}+(J_{16}+B_{16})(J_{16}+J_{17}+B_{17})(J_{16}+J_{17}+J_{18})]+B_{15}B_{17}B_{17}B_{18}C_{18}$$
$$=(J_{15}+B_{15})(J_{15}+J_{16}+B_{16})[J_{15}+(J_{16}+J_{17}+B_{17})(J_{16}+J_{17}+J_{18})]+B_{15}B_{16}B_{17}B_{18}C_{18}$$
$$=(J_{15}+B_{15})(J_{15}+J_{16}+B_{16})(J_{15}+J_{16}+J_{17}+B_{17})(J_{15}+J_{16}+J_{17}+J_{18})+B_{16}B_{16}B_{17}B_{18}C_{18}$$

If $C_{15-18}=(J_{15}+B_{15})(J_{15}+J_{16}+B_{16})(J_{15}+J_{16}+J_{17}+B_{17})(J_{15}+J_{16}+J_{17}+J_{18})$ (4)

and $B_{15-18}=B_{15}B_{16}B_{17}B_{18}$ (5)

then $C_{14}=C_{15-18}+B_{15-18}C_{18}$ (I)

Similarly, we can derive

$$C_{11-14}=(J_{11}+B_{11})(J_{11}+J_{12}+B_{12})(J_{11}+J_{12}+J_{13}+B_{13})(J_{11}+J_{12}+J_{13}+J_{14})$$ (6)

$$B_{11-14}=B_{11}B_{12}B_{13}B_{14}$$ (7)

$$C_{10}=C_{11-14}+B_{11-14}C_{14} \quad \text{(II)}$$

$$C_{7-10}=(J_7+B_7)(J_7+J_8+B_8)(J_7+J_8+J_9+B_9)(J_7+J_8+J_9+J_{10}) \quad (8)$$

$$B_{7-10}=B_7B_8B_9B_{10} \quad (9)$$

$$C_6=C_{7-10}+B_{7-10}C_{10} \quad \text{(III)}$$

$$C_{3-6}=(J_3+B_3)(J_3+J_4+B_4)(J_3+J_4+J_5+B_5)(J_3+J_4+J_5+J_6) \quad (10)$$

$$B_{3-6}=B_3B_4B_5B_6 \quad (11)$$

$$C_2=C_{3-6}+B_{3-6}C_6 \quad \text{(IV)}$$

$$C_{0-2}=(J_0+B_0)(J_0+J_1+B_1)(J_0+J_1+J_2) \quad (12)$$

$$B_{0-2}=B_0B_1B_2 \quad (13)$$

$$C_j=C_{0-2}+B_{0-2}C_2$$

Substituting equations (IV), (III), (II), and (I) successively into equation (V), we obtain

$$C_j=C_{0-2}+B_{0-2}C_{3-6}+B_{0-2}B_{3-6}C_{7-10}+B_{0-2}B_{3-6}B_{7-10}C_{11-14}+B_{0-2}B_{3-5}B_{7-10}B_{11-14}C_{15-18}+B_{0-2}B_{3-6}B_{7-10}B_{11-14}B_{15-18}C_{18} \quad (14)$$

From equations (1), (2), and (4)-(14) and the functions of the basic logic circuits, we can draw the level-by-level comparison circuit logic for comparing 2 19-bit binary numbers Q and X (see Figure).

Using this comparator circuit diagram, if we replace the input X by the input S, the output designation $\overline{Q < X}$ by $\overline{Q \geqslant S}$, and the output designation $Q < X$ by $Q \geqslant S$, we obtain a comparator circuit for 2 19-bit binary numbers Q and S.

The multibit comparator has met design requirements in the course of several years' operation.

8480/9365
CSO: 4008/200

MEMORY ERROR PROCESSING IN 757 COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 3, 1984 pp 1-10

[Article by Zhi Bicen [2388 4310 1478], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] Abstract. This article surveys the processing of memory errors in the 757 computer. In instantaneous-state single-bit errors, processing continues after the error is corrected, while in the case of instantaneous-state multibit errors or address errors, processing of an alternate program is begun. In the case of permanent single-bit errors, either they are corrected and processing continues, or a memory module is replaced and processing continues while error location is carried out for repair purposes. When permanent multibit errors or address errors occur, error location is performed; if a read error is involved, the module is replaced and an alternate program is run, while otherwise an alternate program is run without replacement of the memory module. If both backup memory modules are already in use, even in the case of a read error, the module replacement is not carried out. When there are permanent multibit read errors in all user programs, the system is initialized and disconnection and memory reorganization are carried out with the usable memory modules, after which the machine is restarted.

I. Internal Storage Module Replacement and Disconnection With System Initialized

The 757 vector machine's internal storage has a capacity of 520,000 words in 16 modules (Nos 0-15), together with 2 backup modules (Nos 16 and 17) to replace malfunctioning modules. Figure 1 shows the indicator bits for module disconnection and replacement in general purpose register No 7 (TN7). Bits Nos 49 and 54 are C_{B0} (backup module 1) and C_{B1} (backup module 2): a 1 in either of these locations indicates that the module in question has been substituted for a malfunctioning module. Bits 50-53 and 55-58 give the number of the module replaced by backup modules 0 and 1, respectively. If a module malfunctions after both backup modules have been used, it must be

disconnected, but in order to assure address continuity, at least four modules must be disconnected on each occasion; bits 59 through 62, respectively, indicate disconnections QT_0 (modules 0-3 disconnected when a 1 was present), QT_1 (modules 4-7 disconnected), QT_2 (modules 8-11 disconnected), and QT_3 (modules 12-15 disconnected). The computer can operate in three different disconnection modes:

1. modulo 16: at least 16 modules among Nos 0-17 operating normally;

2. modulo 8 + 4: four modules disconnected (either 0-3, 4-7, 8-11, or 12-15);

3. modulo 8: a maximum of 8 modules can be disconnected, and these must all be within the same group of 8, i.e., either 0-7 or 8-15 may be disconnected in modulo-8 operation.

The memory cannot operate with any combination other than the three named above. Bits 25-42 in TN7 correspond to the on-line switches for modules 0-17. The switches are located on the mainframe instruction control unit control panel. When any of them is depressed, the corresponding bit in TN7 is a 1. When the system is initialized the maintenance personnel must depress the on-line switches for all units which can be on-line, they carry out replacements or disconnections with reference to the switches. These operations are then fanned into TN7. Figure 2 shows the arrangements for the relevant program and the operating system. Figures 3 and 4 are system initialization program flowcharts.

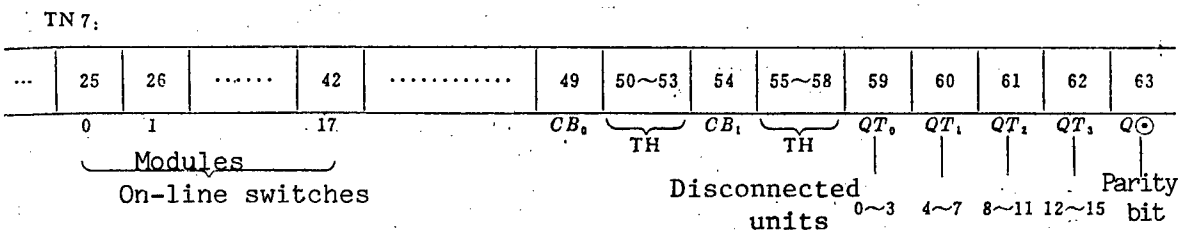


Figure 1.

Disconnection situation				Modular operation	Flag bit (L)
QT_0	QT_1	QT_2	QT_3		
0	0	0	0	16	0
1	1	0	0	8	2
0	0	1	1		
1	0	0	0	8+4	1
0	1	0	0		
0	0	1	0		
0	0	0	1		
Other combinations				not used	3

Figure 2.

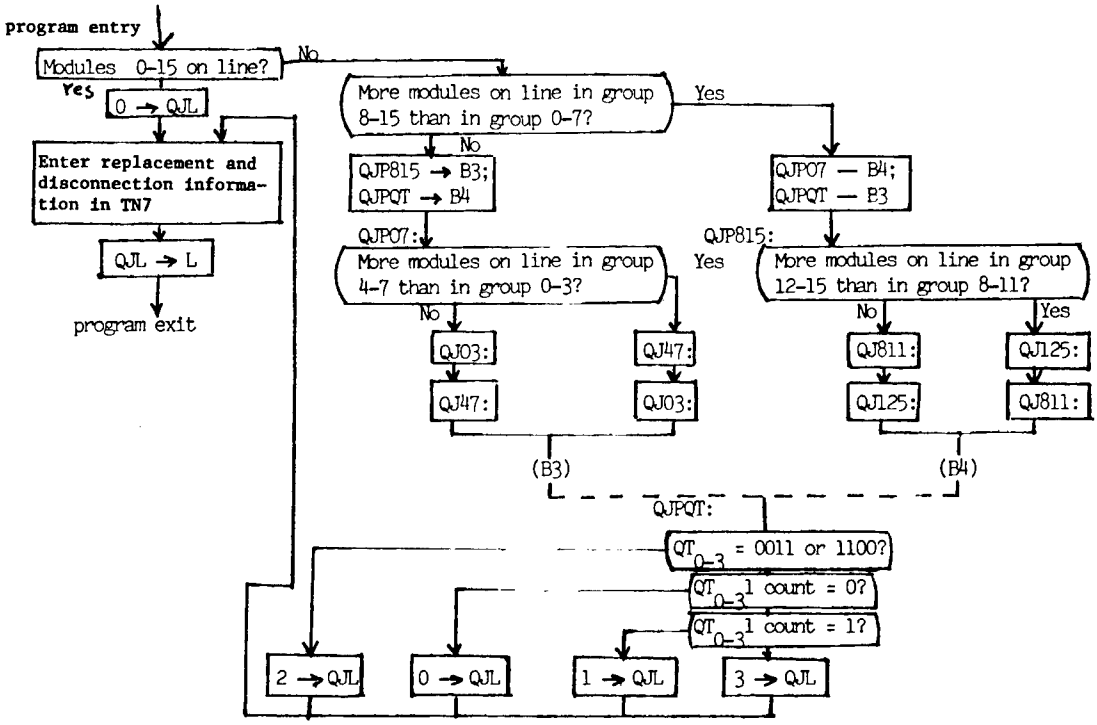


Figure 3.

II. Error Logging Table

Each time an error occurs, the type of error, the number of the module in which it occurs, the number of the incorrect bit, whether it is an intermittent or permanent fault, and the like, are recorded in the error logging table for reference. The error logging table occupies 512 locations (64 bits each) in page 63 of the peripheral processor memory; each time these 512 locations are filled, the contents are transferred to tape, after which the 512 locations are cleared and logging starts again at the beginning. The first location gives the year, month, day, hour, minute, and second of the error, and the time indicator (1, 1, 1) is set in bits 1-3. The error record proper is placed in the next location. When the next error is to be recorded, if the year, month, and day are the same as for the preceding error, then the error is logged in the next location; otherwise the time indicator is placed in the next location and the error record proper in the location following it. The log format is as follows:

Time indicator:

1 1 1 ·····	Year	Month	Day	Hour	Minute	Second
0 1 2 3 ~ 9	10 ~ 25	26~33	34~41	42~49	50~57	58~63

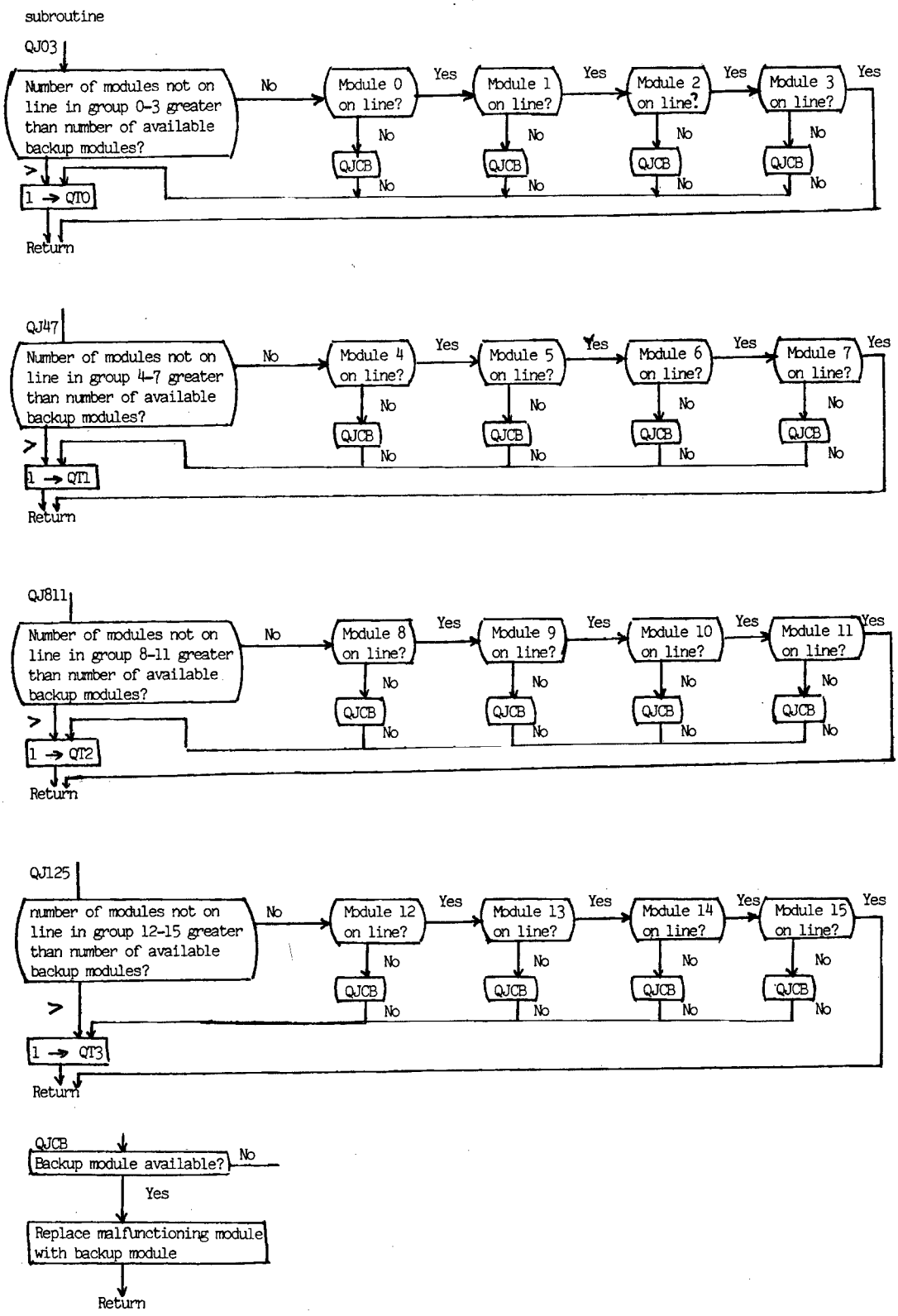

subroutine
QJ03
Number of modules not on line in group 0-3 greater than number of available backup modules?
No
Module 0 on line?
Yes
Module 1 on line?
Yes
Module 2 on line?
Yes
Module 3 on line?
Yes
No
QJCB
No
>
1 → QT0
Return
QJ47
Number of modules not on line in group 4-7 greater than number of available backup modules?
No
Module 4 on line?
Yes
Module 5 on line?
Yes
Module 6 on line?
Yes
Module 7 on line?
Yes
No
QJCB
No
>
1 → QT1
Return
QJ811
Number of modules not on line in group 8-11 greater than number of available backup modules?
No
Module 8 on line?
Yes
Module 9 on line?
Yes
Module 10 on line?
Yes
Module 11 on line?
Yes
No
QJCB
No
>
1 → QT2
Return
QJ125
number of modules not on line in group 12-15 greater than number of available backup modules?
No
Module 12 on line?
Yes
Module 13 on line?
Yes
Module 14 on line?
Yes
Module 15 on line?
Yes
No
QJCB
No
>
1 → QT3
Return
QJCB
Backup module available?
No
Yes
Replace malfunctioning module with backup module
Return

Figure 4.

Error record:

Control number	Error class	Error type	Replacement situation	Replacement number	Address error bit: Multi-bit error	Address error bit: Error bit 1	Address error bit: Error bit 2	Message display	Number of double calculations	Time of error	Recompute time
0~2	3~5	6~8	9	10	11~15	16	17~23	24~30	31~32	33~35	36~55	56	63

Control number: 100, memory error; 001, memory control unit error.

Error class: 001, internal store address error; 010, internal store data code double error; 011, internal store data code single error.

Error type: 100, address error; 001, write error; 011, single read error; 110, multibit read error.

Replacement situation: 00, no replacement: 01, replacement module 0 used; 10, replacement module 1 also used.

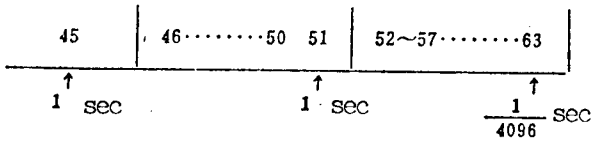
Error bit: when there is an internal storage address error, a 1 is placed in the corresponding position (0 to 14) to indicate bit 0 through 14. Two wrong bits can be recorded for a data code error: error bit 1 is recorded in positions 17-23, and wrong bit 2 in positions 24-30 (in binary form). If there are more than 2 wrong bits, then a 1 is placed at position 16.

Jitter: 00, jitter; 01, permanent fault, display message.

Number of double calculations: a number from 0 to 7; 7 indicates a permanent fault.

Time of fault: contains bits Nos 44-63 in time indicator.

Double calculation time: contains bits 50-57 of peripheral processor clock, indicating the following:



If the double calculation is too long for the sapce, then all zeroes are recorded.

The format of the signals sent to the operating system when there is an internal storage module error is as follows:

Fault class \ Bits		0	1	2	3	4~18	19~23
Internal store multi-bit	One location	0	1	1	1	Error address	Unit in which error occurs
	More than one location	1	1	1	1	"	"
		0	1	0	0	"	"

III. Diagnosis and Processing of Internal Storage Address Errors

When an address errors, the message "NCDZCO" is produced in the memory control unit; the program recovers the number of the malfunctioning module and the address where the error occurred from the memory control unit and accesses this unit and the complemented address of the error (internal storage read); it then recovers the internal storage address and determines in which bit or bits the error occurred, waits 200 milliseconds, and accesses the module again. If the error disappears in fewer than seven accesses, then this is regarded as a temporary error; it is logged and a jump is made to the "nonrecomputable" entry so that an alternate program can be run. If the fault still persists after seven accesses, this is regarded as a permanent fault. It is logged and the following message is displayed in the following format (screen display):

```
NCGUZHA: XXXX. XX. XX. XX. XX. XX,
     COL: NCDZCO;
    COTH: X;
     COW: X; X; ·····X;
  DENGDAIWEIXIU; X SPP;
```

This is interpreted as follows: NCGUZHA: year, month, day, hour, minute, second. COL, error class: NCDZCO, internal storage address error. COTH: number of malfunctioning module, X. COW, bit in which error occurred. DENGDAIWEIXIU, awaits servicing. X, SPP, message displayed X times.

The maintenance personnel carry out the repairs indicated in the message, then type in a command on the supervisor console to return to the test program entry. If the test results indicate that the error is still present, the message is displayed again. If testing indicates no error is present, a jump is made to the "nonrecomputable" entry and an alternate program is run.

IV. Diagnosis and Processing of Single Error in Data Code

If there is a parity error in a data code during a read or write operation, then an internal storage read error (NCDCO) or internal storage write error (NCXCO) message is emitted; during a read from internal storage, if there is a "read error stop," a data code single error (SMDCO) message is sent by the memory control unit*. When the internal storage produces a read error or write error message the machine is not stopped; after the memory control unit has emitted a data code single error message it stops the machine and runs an alternate program. It first mus be determined whether a write error or a read error is involved, and whether it is a single-bit or multibit error. In the case of either a read or write error, provided that it is only a single-bit error,

*"Single error stop" can be set by switch or by the program. When the machine is not stopped for a single error, the error is either corrected by hardware or tolarated. Because a single-bit error may include a multibit odd-parity error, the problem of a wrong parity check arises. In addition, because of failure to eliminate a single-bit fixed error, it is fairly easy for a multibit error to arise.

the mainframe is restarted and processing continues after fault processing has been carried out.

A data code single error is a memory control unit error, and when it occurs the memory control unit is stopped in that same beat, while the instruction controller and operation control unit are stopped in the next beat, so that the look-ahead register (X), look-behind register (H), index buffer register (BH), and general register (TN), which are involved with the memory control unit, all proceed for 1 beat longer than it. In order to be able to start the machine and continue operation after error processing, it is necessary to carry out start-stop processing as follows.

1. Memory control unit start-stop processing*

① Recover state.

② Determine which instruction is in memory control unit station 6 (JM 6). If it is NC → X, or Y → X, or Yb → X, or CK → X, or NC → TN, or NC → BH, or NC → ZH, then a ZOCJM 6 pulse is emitted and this instruction is discarded.

③ The instruction in memory control unit stations 1 and 2 (JM 1, JM 2) are determined. If the instruction in JM 1 is an effective H → Yb instruction and T200 = 1, or if the instruction in JM 2 is an effective H → NC, or H → Q, or H → QY, or H → Y, or H → BH, and T300 = 1, then, depending on MD, the following are emitted:

i) when T_B = 1, then pulses Z 1 CYHi and Z 1 CZHi are emitted; ii) when T_R = 0, then ZOCKH, Z 1 CZHi, and Z 1 CYHi are emitted and 15 DRHi and DRJSHi pulses are emitted to back the look-behind register up 1 beat. Then, if CKH = 1, a Z 1 CKH pulse is emitted, while if CKH = 0, then ZOCKH is emitted (i = 0, 1).

④ If JM 2 contains an effective NC → X instruction and CJSX = 1, then depending on MD, 15 DRJSXi pulses are emitted to back the look-ahead register up 1 beat (i = 0, 1, 2, 3).

⑤ If JM 2 contains an effective BH → NC instruction and T300 = 1, then dependig on MD, 15 DRBHi and JSBHi + 1 pulses are emitted to back the index buffer register up 1 beat (i = 0, 1, 2, 3).

⑥ If JM 2 contains an effective TN → NC instruction and T300 = 1, then 15 DRTD pulses are emitted to back the general purpose register up 1 beat.

⑦ The instruction in memory control unit station 3 (JM 3) is determined. If it is H → BH, then a ZOCJM 3 pulse is emitted to discard this instruction.

2. Processing of External Instructions

If an internal instruction is involved when the internal storage read error message is emitted (i.e., an instruction to exchange data with peripheral

*See articles on the 757 instruction set and the mainframe control units.

processor channels 0 or 1 or disk channels 0 or 1), a signal must be sent to the operating system so that the requisite processing can be performed.

For an NC $\rightleftharpoons$ WP_0 [internal storage and peripheral processor channel] instruction, a flat 3 is sent;

For an NC $\rightleftharpoons$ WP_1 instruction, a flag 4 is sent.

For an NC $\rightleftharpoons$ P_0 [disk channel] instruction, a flag 5 is sent.

For an NC $\rightleftharpoons$ P_1 instruction, a flag 6 is sent.

3. Processing of Write Errors

When reading from main memory, once the memory control unit sends a data code single error (SMDCO) message, it is first determined whether a write error is involved. The module number and address number in which the error occurred are recovered and sent to stations 3 and 5 (JM 3, JM 5) of the memory control unit, and the data code (all 1's or all 0's) is distributed to rewrite register (JCXS) bit Nos 0-71 (Nos 0-63 are the data code bits, and 64-71 are the check bits). Then an internal storage write access command is issued and the data code resulting from the write operation is recovered and examined for errors. If there is a write error, after a delay of 200 milliseconds another write operation is attempted. If the error disappears in fewer than seven accesses, it is again determined whether a read error exists. If a write error is still present after seven accesses, then it is a permanent fault; a determination is made whether it is a 1-bit or multibit fault, the fault is logged and a message is displayed.

(1) If it is a 1-bit fault, the message is as follows:

```
NCGUZA: XXXX. XX. XX. XX. XX. XX,
    COL: NCSMDCO: CK→NCTDCO\NCXSJCO;
  COTH: X,
   COW: X,
  DENGDAIWEIXIU; XSPP;
```

Other than COL [fault class], the display is the same as for an address error. In COL, NCSMDCO indicates an internal storage data code single fault; CK → NCTDCO/NCXSJCO indicates memory control unit → internal store channel fault or data write register fault.

The maintenance personnel perform the indicated repairs, then type in a command on the supervisor console to return to the test program entry. If the test results indicate no error, then the status is restored and memory control unit start-stop processing is executed, the mainframe is started, and processing continues. If the test indicates that a write error still exists, the message is displayed again.

(2) In the case of a multibit fault, the display is as follows:

```
NCGUZA: XXXX. XX. XX. XX. XX. XX,
   COL: NCSMDOCO: CK→NCTDCO\NCXSJCO;
  COTH: X;
   COW: X, X, ·····X,
  DENGDAIWEIXIU; XSPP;
```

Other than COL, this is the same as for an address fault. Under COL, NCSMDOCO indicates internal storage data code multiple fault. CK → NCTDCO/ NCXSJCO indicates internal storage memory control unit → internal storage channel error.

The maximum number of bit-error indications [COW] that can be displayed is 72. The maintenance personnel perform the repairs indicated, then type in a command on the supervisor console to return to the test program entry. If the test indicates no error, a jump is made to the "nonrecomputable" entry. If the test still indicates a fault, the message is displayed again.

4. Processing of Read Errors

Once it is determined that there is no write error, all 1's are written into the internal store and a read command sent; then all 0's are written in and a read command sent. If the data codes that are read out are found to contain an error, then after a 200-millisecond delay another read from internal store command is sent. If the error disappears within seven accesses (this is the recompute number, which includes the total number of internal storage accesses for write and read) it is treated as a temporary error and logged, after which the machine status is restored and start-stop processing executed, after which the mainframe is started and operation continues. If the error is still present after seven accesses, it is necessary to determine whether it is a 1-bit or multibit read error.

(1) One-bit read error: it is necessary to determine whether a backup memory module is available and on line.

(i) No backup module available or no module on line.

After the error is logged, the following message is displayed:

```
NCGUZA: XXXX. XX. XX. XX. XX. XX,
   COL: NCTDCO;
  COTH: X;
  CODZ: XXXXX,
   COW: X;
  BTWAN; JINJIDAIXIU;
```

Differences from earlier message displays are as follows. In line 2, NCTDCO indicates internal storage module single error; in line 4, CODZ indicates the error address; in line 6, BTWAN indicates backup modules all in use, and JINJIDAIXIU indicates repair urgently needed.

After this message is displayed the return is made automatically without the need to type in a command, the state is restored and the mainframe continues operation. Because the single error is not yet removed, "single error no stop" must be set; the hardware corrects or tolerates the fault.

(ii) Backup module is available and on line:

The contents of the malfunctioning module, with the error corrected, are entirely transferred to the backup module, which then replaces the malfunctioning module; the latter is switched out and the maintenance personnel make the necessary repairs with it off line.

When the module is replaced, the number is entered in TN 7, the error is logged, and the following message is displayed:

```
NCGUZHA: XXXX. XX. XX. XX. XX. XX;
    COL: NCTDCO;
   COTH: X;
   CODZ: XXXXX;
    COW: X;
 TOJIJIANXIU;
```

This display differs from the preceding ones in line 6, where TOJIJIANXIU indicates off-line repair.

After this message is displayed, the return is automatic; the machine state is restored, "single error stop" is set (so that if another single error occurs it will still be possible to stop the machine), and the mainframe is restarted. The maintenance personnel switch off the on-line switch for the affected unit.

(2) Multibit error: Although the mainframe cannot continue operation after a multibit error is processed, if a multiple-location error is not involved then other parts of the malfunctioning module can still be used and an alternate program can be run; accordingly a determination must be made whether a multi-location error is involved and the operating system must be notified of the module number, the address of the error, and whether it is a single-location or multiple-location error.

(i) Single-location error. It is determined whether a backup module is available and on line.

i) If no backup module is available or none is on line, the following message is displayed:

```
NCGUZA: XXXX. XX. XX. XX. XX. XX;
    COL: NCTDOCO;
   COTH: X;
   CODZ: XXXXX;
    COW: X; X; ·····X;
 BTWAN; JINJIDAIXIU;
```

This differs from the preceding messages as follows. In line 2 [error class], NCTDOCO indicates an internal storage module multiple fault; the bit number given is full word length, a maximum of 72 bits. An automatic return is made to the "nonrecomputable" entry and an alternate program is run.

2) Backup module available and on line

The contents of the malfunctioning unit are entirely transferred to the back-up unit, which then replaces the malfunctioning unit. The malfunctioning unit is repaired off line. After the replacement, the fault is logged and the following message is displayed:

```
NCGUZHA: XXXX. XX. XX. XX. XX. XX;
    COL: NCTDOCO;
   COTH: X;
   CODZ: XXXXX;
    COW: X; X; ·····X;
 TOJIJIANXIU;
```

After the message is displayed, the return is made automatically; a jump is made to the "nonrecomputable" entry and an alternate program is run (because even though there is no error in the backup unit that has been substituted, the contents that were transferred contain a one-address multibit error which cannot be corrected, so that the only course is to run an alternate program). The maintenance personnel switch off the on-line switch for the malfunctioning module.

(ii) Multi-location error

The error is logged and the operating system is notified of an internal storage multibit error and multiple-location error. The following message is displayed:

```
PPDOCO:
NCGUZA: XXXX. XX. XX. XX XX. XX;
   COL: NCTDOCO;
  COTH: X;
  CODZ: XXXXX;
   COW: X; X; ·····X;
 NCDODYCO;
```

The difference from the preceding displays is as follows. In the sixth line, NCDODYCO indicates a multi-location error. The return is automatic after this message is displayed: a jump is made to the "nonrecomputable" entry and processing is carried out by the operating system.

V. Diagnosis and Processing of Data Code Double Errors

If during reading from main memory the data code has an error in an even number of bits, then the memory control unit produces the data code double

error message (SMSCO). The diagnosis and processing of this error are the same as for a multibit data code single error.

VI. Processing After Repair of Malfunctioning Modules

1. If the malfunctioning module which has been taken off line for repair is one of Nos 0-15, after it has been repaired a suitable time during computation is chosen, the relevant on-line switch on the control panel is depressed, and an operator emergency notification is used to temporarily halt program operation. An "internal store repaired, restore" command is typed in on the supervisor console and the repaired unit is transferred to the program's module diagnosis (ZDT) entry. It is first tested to determine whether it actually is repaired.

(1) If the module is found to still have a fault, the following is displayed:

```
PCO:
NCGUZHA: XXXX. XX. XX. XX. XX. XX,
    COL: XIUTCO;
   COTH: X,
   CODZ: XXXXX,
    COW: X, X, ·····X,
```

The difference from the preceding displays is as follows. In the second line (error class), XIUTCO indicates error in repaired module. The return is made automatically after this message is displayed; the state is restored and the mainframe is restarted and continues operation.

(ii) If the test finds no faults, then the contents of the backup module which is replacing this module are entirely transferred back to the original module, which is then put on line in place of the backup module, the state is restored, and computation continues.

2. If the module which has been repaired off-line is one of the backup modules (No 16 or 17), when repair is completed a suitable time in the course of operation is chosen, the relevant on-line switch is depressed on the control panel, and an operator emergency notification is used to temporarily stop operation. Then the "module repaired, restore" command is entered on the supervisor console and the number of the repaired unit is sent to the module diagnosis (ZDT) entry of the relevant program, in order to determine whether the module has been properly repaired.

(1) If the module is found to still have an error, the situation is handled as in the case of units 0-15, a message is displayed and computation continues.

(ii) If the module is found to have no errors, then the "single error stop" flag is set at 0 (so that if another single error arises during the computation the machine can be stopped and a branch made to the diagnostic program, since a backup is now available for replacement), after which the machine's state is restored and the mainframe is restarted.

3. After an operator emergency notification is entered for a repaired memory module, if the module number for the "module repaired, restore" command is typed in incorrectly on the supervisor console, the following occurs:

(i) if the number is that of a module from 0 to 15 and that has not been replaced by a backup module,

(ii) or if the number typed in is that of a backup module which is currently replacing some other module, then the following message is displayed:

MLCO:

MLCO

MLCO indicates a command error. After it is displayed a return to computation is made automatically.

VII. Program Flowcharts

Figures 5-9 are program flowcharts for the processing of internal storage errors.

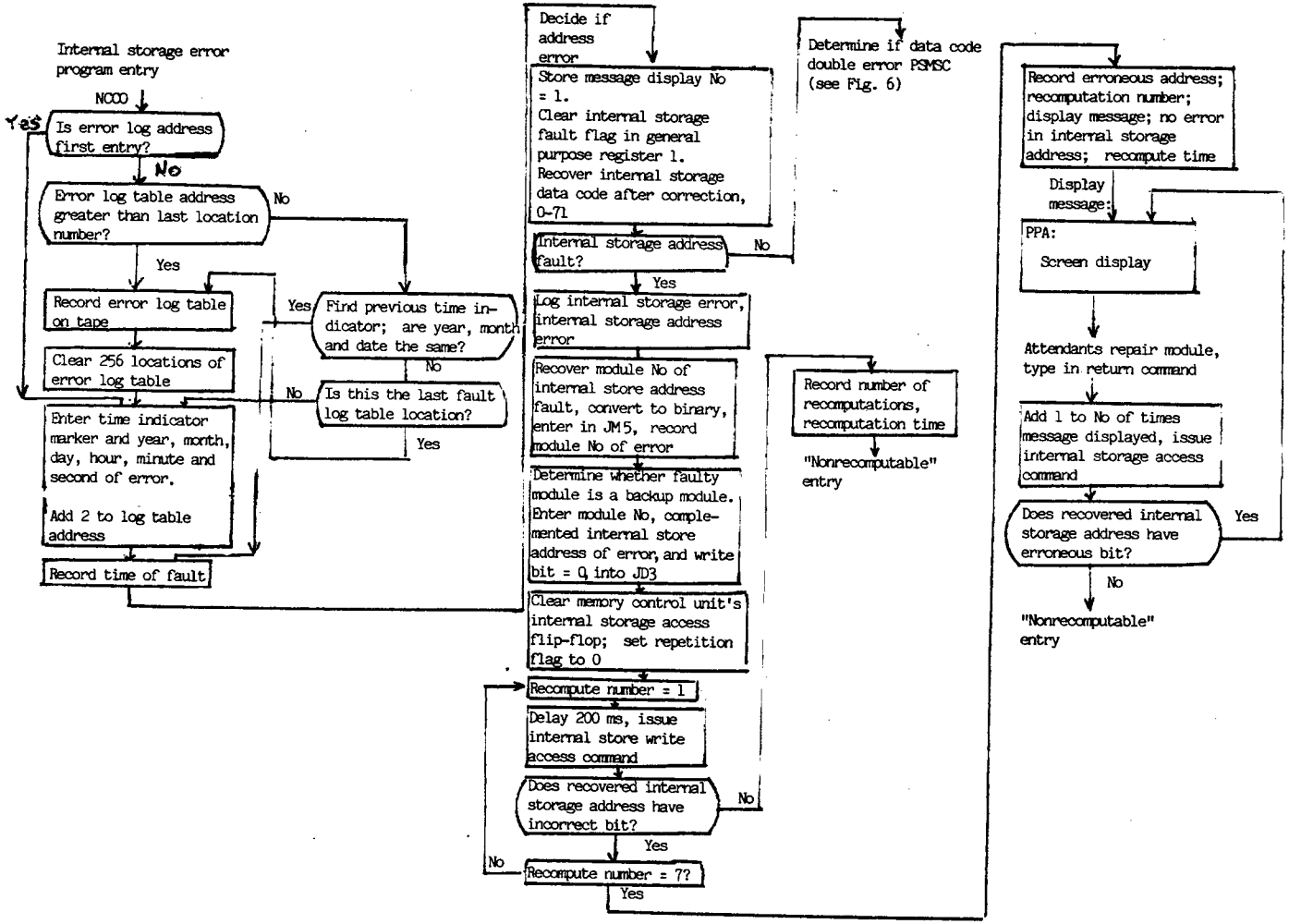

Internal storage error program entry
NCCD
Is error log address first entry?
Yes
No
Error log table address greater than last location number?
No
Yes
Record error log table on tape
Clear 256 locations of error log table
Enter time indicator marker and year, month, day, hour, minute and second of error.
Add 2 to log table address
Record time of fault
Yes
Find previous time indicator; are year, month and date the same?
No
Is this the last fault log table location?
No
Yes
Decide if address error
Store message display No = 1.
Clear internal storage fault flag in general purpose register 1.
Recover internal storage data code after correction, 0-71
Internal storage address fault?
No
Yes
Log internal storage error, internal storage address error
Recover module No of internal store address fault, convert to binary, enter in JM5, record module No of error
Determine whether faulty module is a backup module. Enter module No, complemented internal store address of error, and write bit = 0 into JD3
Clear memory control unit's internal storage access flip-flop; set repetition flag to 0
Recompute number = 1
Delay 200 ms, issue internal store write access command
Does recovered internal storage address have incorrect bit?
No
Yes
Recompute number = 7?
No
Yes
Determine if data code double error PSMSC (see Fig. 6)
Record number of recomputations, recomputation time
"Nonrecomputable" entry
Record erroneous address; recomputation number; display message; no error in internal storage address; recompute time
Display message:
PPA:
Screen display
Attendants repair module, type in return command
Add 1 to No of times message displayed, issue internal storage access command
Does recovered internal storage address have erroneous bit?
Yes
No
"Nonrecomputable" entry

Figure 5.

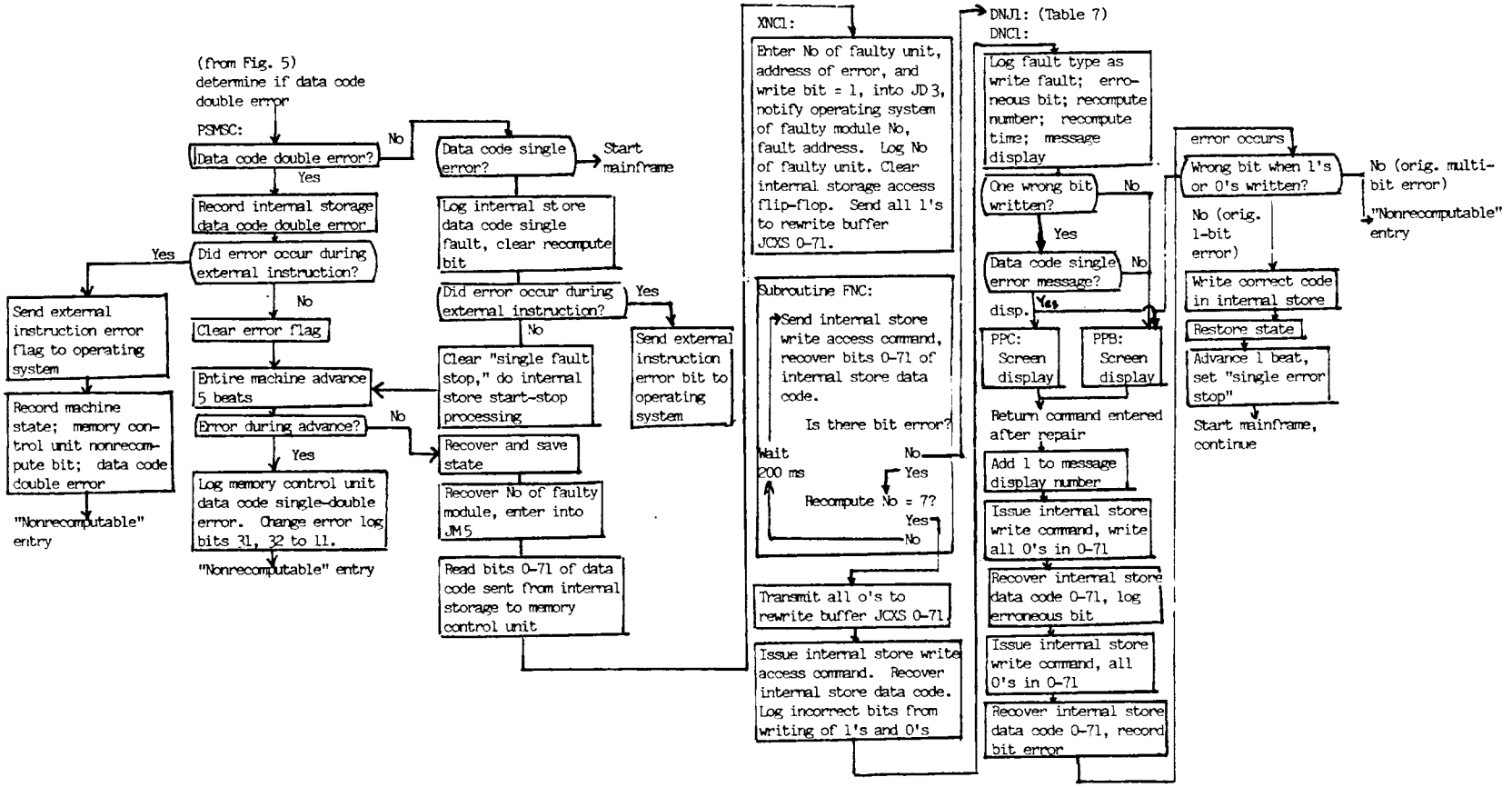

(from Fig. 5)
determine if data code double error
PSMSC:
Data code double error?
No
Yes
Data code single error?
Start mainframe
Record internal storage data code double error
Log internal store data code single fault, clear recompute bit
Yes
Did error occur during external instruction?
Did error occur during external instruction?
Yes
No
Send external instruction error flag to operating system
Clear error flag
Send external instruction error bit to operating system
No
Clear "single fault stop," do internal store start-stop processing
Entire machine advance 5 beats
Error during advance?
No
Recover and save state
Yes
Record machine state; memory control unit nonrecompute bit; data code double error
Log memory control unit data code single-double error. Change error log bits 31, 32 to 11.
Recover No of faulty module, enter into JM5
"Nonrecomputable" entry
"Nonrecomputable" entry
Read bits 0-71 of data code sent from internal storage to memory control unit
XNC1:
Enter No of faulty unit, address of error, and write bit = 1, into JD3, notify operating system of faulty module No, fault address. Log No of faulty unit. Clear internal storage access flip-flop. Send all 1's to rewrite buffer JCXS 0-71.
Subroutine FNC:
Send internal store write access command, recover bits 0-71 of internal store data code.
Is there bit error?
Wait 200 ms
No
Yes
Recompute No = 7?
Yes
No
Transmit all o's to rewrite buffer JCXS 0-71
Issue internal store write access command. Recover internal store data code. Log incorrect bits from writing of 1's and 0's
DNJ1: (Table 7)
DNC1:
Log fault type as write fault; erroneous bit; recompute number; recompute time; message display
One wrong bit written?
No
Yes
Data code single error message?
No
Yes
disp.
PPC: Screen display
PPB: Screen display
Return command entered after repair
Add 1 to message display number
Issue internal store write command, write all 0's in 0-71
Recover internal store data code 0-71, log erroneous bit
Issue internal store write command, all 0's in 0-71
Recover internal store data code 0-71, record bit error
error occurs
Wrong bit when 1's or 0's written?
No (orig. multi-bit error)
"Nonrecomputable" entry
No (orig. 1-bit error)
Write correct code in internal store
Restore state
Advance 1 beat, set "single error stop"
Start mainframe, continue

Figure 6.

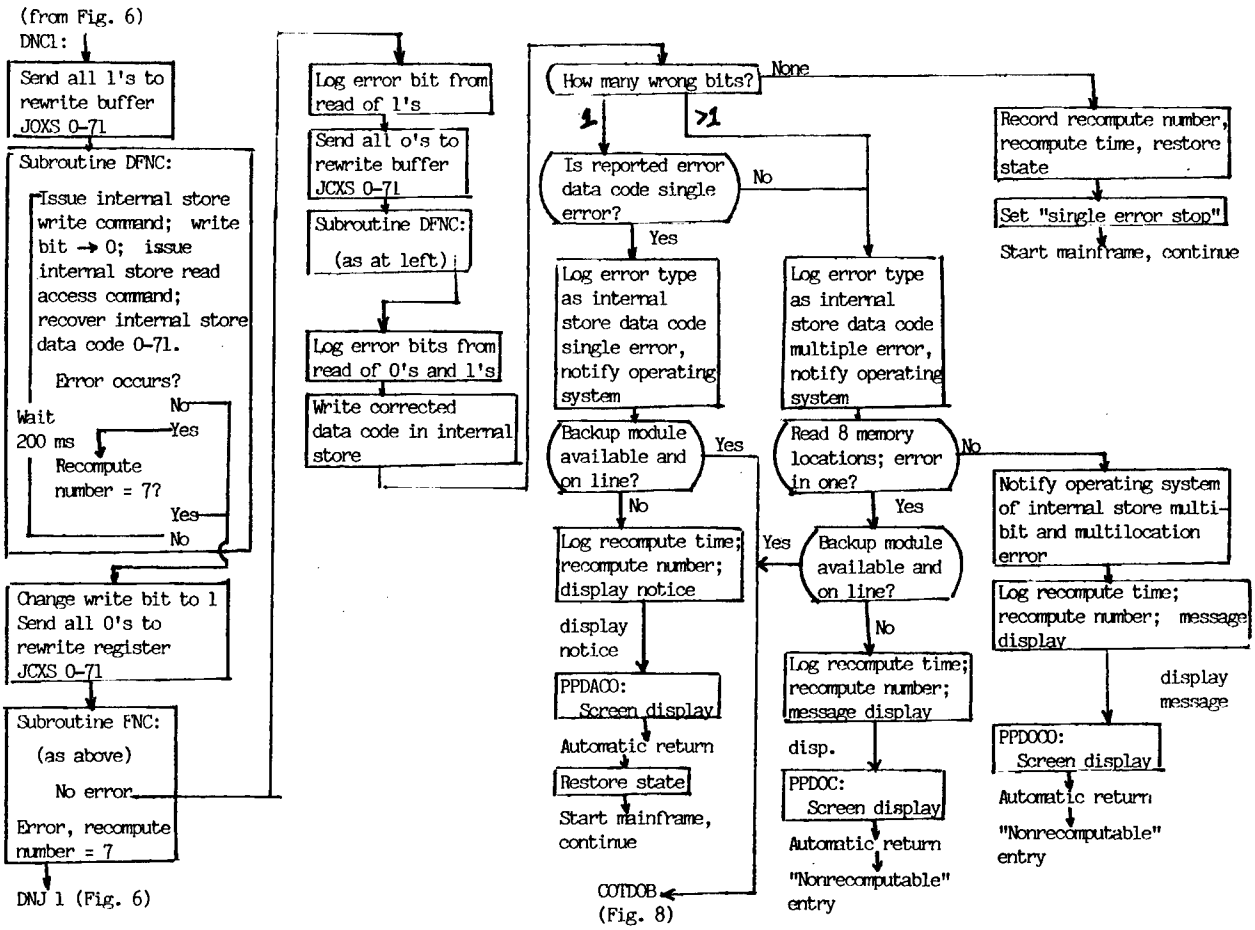
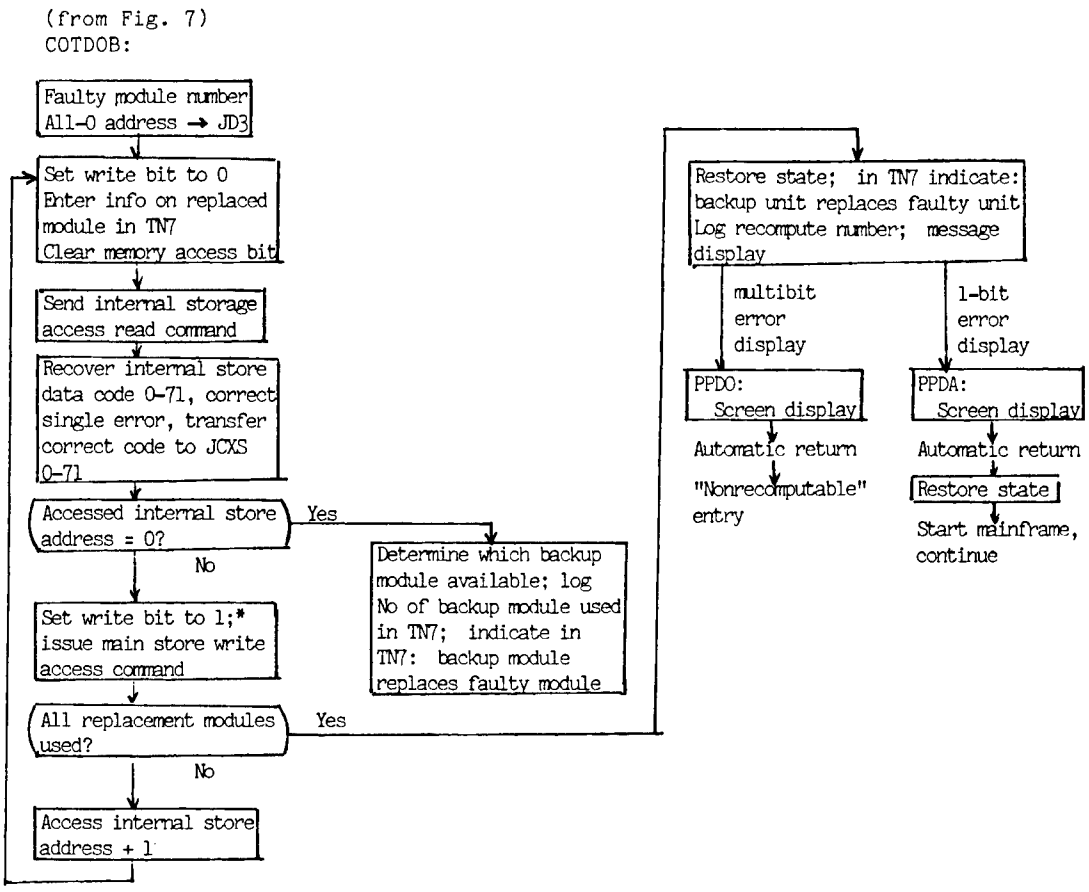

(from Fig. 6)
DNC1:
Send all 1's to rewrite buffer JOXS 0-71
Subroutine DFNC:
Issue internal store write command; write bit → 0; issue internal store read access command; recover internal store data code 0-71.
Error occurs?
No
Yes
Wait 200 ms
Recompute number = 7?
Yes
No
Change write bit to 1
Send all 0's to rewrite register JCXS 0-71
Subroutine FNC:
(as above)
No error
Error, recompute number = 7
DNJ 1 (Fig. 6)
Log error bit from read of 1's
Send all o's to rewrite buffer JCXS 0-71
Subroutine DFNC:
(as at left)
Log error bits from read of 0's and 1's
Write corrected data code in internal store
How many wrong bits?
None
1
>1
Is reported error data code single error?
No
Yes
Log error type as internal store data code single error, notify operating system
Backup module available and on line?
Yes
No
Log recompute time; recompute number; display notice
display notice
PPDACO:
Screen display
Automatic return
Restore state
Start mainframe, continue
COTDOB
(Fig. 8)
Log error type as internal store data code multiple error, notify operating system
Read 8 memory locations; error in one?
No
Yes
Backup module available and on line?
Yes
No
Log recompute time; recompute number; message display
disp.
PPDOC:
Screen display
Automatic return
"Nonrecomputable" entry
Record recompute number, recompute time, restore state
Set "single error stop"
Start mainframe, continue
Notify operating system of internal store multi-bit and multilocation error
Log recompute time; recompute number; message display
display message
PPDOCO:
Screen display
Automatic return
"Nonrecomputable" entry

Figure 7.



* Indicate backup unit replaces faulty module in TN7

Figure 8.

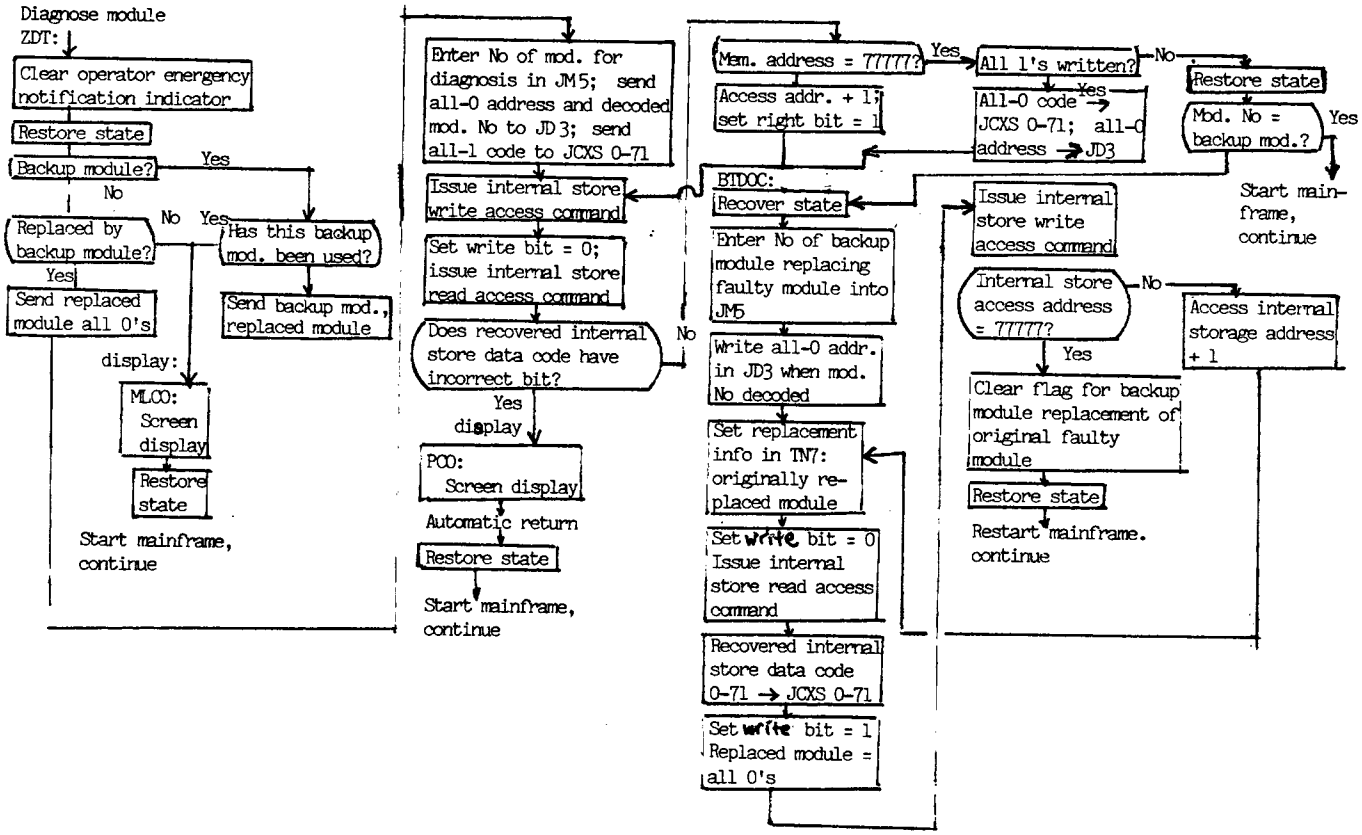

Diagnose module ZDT:
Clear operator emergency notification indicator
Restore state
Backup module?
Yes
No
Replaced by backup module?
No
Yes
Has this backup mod. been used?
Yes
Send replaced module all 0's
Send backup mod., replaced module
display:
MLCO: Screen display
Restore state
Start mainframe, continue
Enter No of mod. for diagnosis in JM5; send all-0 address and decoded mod. No to JD3; send all-1 code to JCXS 0-71
Issue internal store write access command
Set write bit = 0; issue internal store read access command
Does recovered internal store data code have incorrect bit?
No
Yes
display
PCO: Screen display
Automatic return
Restore state
Start mainframe, continue
Mem. address = 77777?
Yes
All 1's written?
No
Yes
Restore state
Access addr. + 1; set right bit = 1
All-0 code → JCXS 0-71; all-0 address → JD3
Mod. No = backup mod.?
Yes
Start mainframe, continue
BITDOC:
Recover state
Enter No of backup module replacing faulty module into JM5
Write all-0 addr. in JD3 when mod. No decoded
Set replacement info in TN7: originally replaced module
Set write bit = 0
Issue internal store read access command
Recovered internal store data code 0-71 → JCXS 0-71
Set write bit = 1
Replaced module = all 0's
Issue internal store write access command
Internal store access address = 77777?
No
Yes
Access internal storage address + 1
Clear flag for backup module replacement of original faulty module
Restore state
Restart mainframe. continue

Figure 9

THERMAL DESIGN OF 757 COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER SCIENCE AND DEVELOPMENT] in Chinese Vol 21 No 3, 1984 pp 11-20

[Article by Tang Mali [0781 3854 7787], Tang Zhongchai [3282 6988 2088], and Li Minwang [7812 3046 2598], Institute of Computing Technology, Chinese Academy of Sciences]

[Excerpts] Abstract. This article compares the advantages and disadvantages of long-path and short-path forced-air cooling of computers. It briefly describes the three types of short-path ventilation design used in the 757 machine, simulation experiments, test results on the actual machine frames, the advantages and disadvantages of the methods, and matters to which attention must be directed when using this technique.

II. Thermal Design of the 757 Computer

The principles that were followed in the thermal design of the 757 machine are as follows:

(1) High reliability with a maximum junction temperature in IC packages not exceeding 85°C (maximum junction temperature for 500 mW bipolar semiconductor internal storage devices not exceeding 100°C), maximum junction temperature difference not exceeding 25°C; (2) simple construction, convenience of maintenance and management; (3) low noise and comfortable temperature in the computer room; and (4) subject to fulfillment of the above conditions, use of the smallest cooling system possible, in order to decrease production cost.

Choice of heat exchange method: Excellent cooling results can be obtained with conductive cooling using water or freon, but the design is complex, there are stringent requirements regarding machining precision, maintenance and management are complex, and manufacturing costs are high, so that this method was not used. Since the computer uses small-scale integration [SSI] circuitry, the individual packages have low power consumption, and convective forced-air cooling will meet heat dissipation requirements.

Choice between centralized or decentralized ventilation. Frames with centralized ventilation need not be equipped with blowers, which decreases noise in the computer room and makes it possible to decrease frame dimensions; in addition, relatively low temperature air can be fed directly into the frames, then, after cooling, allowed to escape into the machine room, so that the frames can operate at a relatively low temperature and the machine room will be comfortable.

Choice between long and short air paths: Short air path ventilation involves a simple design, there is no need to seal the front areas of the cards (and it is relatively easy to seal the air intake space), there are no air short circuits, uniform ventilation is easily achieved, and the maximum junction temperature and junction temperature difference are relatively low; in addition, because SSI circuits are used and the package power consumption is low, heat dissipation requirements can be met without high air speed. Choice between open- and closed-path systems. Air return ducts are not needed in an open-path system, which makes for convenience in layout and frame placement.

Based on the above considerations and the types of cards and boards used, we designed three short-path centralized open-path ventilation methods (decentralized ventilation was used for the peripheral devices, and each frame was equipped with a blower).

1. Thermal Design of Mainframe Internal Core Storage Frame, Peripheral Processor Operation Control Unit and Channel Frame, and Peripheral Device Controller Frame

The ventilation design shown in Figure 2 was used. Cold air passes from the bottom of the frame into the air intake space between two boards, then reaches the cards via air intake slots on both sides of the card connectors. After cooling the components it exits through the front of the card area and reaches the machine room through slots around the frame door. The heat in the power supply drawers is dissipated by sending air through holes in the air passage opposite the heat-sensitive parts and high-power consumption parts, after which it exits through the top of the frame.

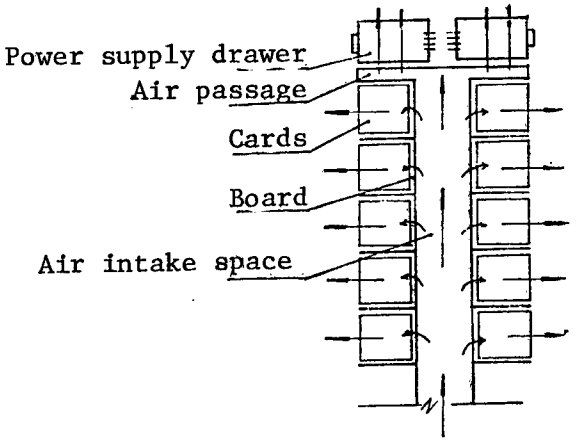


Figure 2. Short Air Path Ventilation, Type 1

Example: The thermal design of the mainframe core storage frame.

Each frame contains 92 circuit cards, each of which consumes 7 w; the cards are 26 mm apart, and the slots on both sides of the card connectors are 0.5 mm wide.

Each frame contains 100 control cards and amplifier cards, each of which consumes 2 w; the cards are spaced 17 mm apart, and the slots on both sides of the card connectors are 0.25 mm wide.

Each frame has three large power supply drawers, each with a power consumption of 110 w: 12 holes 12 mm in diameter are provided for each drawer.

Each frame has six small power supply drawers and each drawer is provided with five air holes 8 mm in diameter.

Calculation results indicate that when the static pressure in the air intake space is 2.2 mm H_2O, the flow rate in each frame is 400 m^3/hour (the calculation procedure will not be described). The system has 18 vector processor internal core storage frames, which together take 7,200 m^3/hr of air.

The ratio of the total area of card intake air slits and power supply drawer intake air holes Σf to the cross sectional area of the air intake space is $\frac{\Sigma f}{F} \approx 12$ percent; because this ratio is small, the speed of the air exiting through these slits and holes is uniform.

The advantages of this ventilation method are: 1) each card is uniformly ventilated; 2) the ventilation method is simple and easy to implement, and a large amount of ventilation hardware is not required; 3) the front section of the cards does not need to be sealed, which is convenient for machine adjustment and maintenance.

The disadvantages are:

(1) owing to the limited number of card modules, there is only a limited number of air intake hole sizes, and accordingly air intake slits are used for cards with different power consumption levels, so that their temperature rises differ at a given air flow,

(2) the widths of the slots must be made as small as possible in order to assure uniform airflow, and accordingly they may become blocked when dust collects on the cards, resulting in excessively high temperatures in the frames if they are not cleaned in time,

(3) when printed circuit boards are used, air slots cannot be placed on both sides of the card connectors, and thus this method cannot be used.

2. Thermal Design of the Peripheral Processor Core Storage Frame

The short air path ventilation passage design shown in Figure 3 is used.

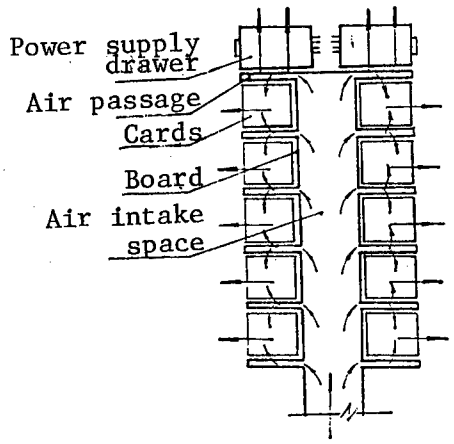


Figure 3. Short Air Path Ventilation, Type 2

The cards used in the core storage of the peripheral processor are large, measuring 520 x 480 mm. Three large cards form one 6-bit subunit, and 12 subunits form one 72-bit unit; each frame contains two 72-bit units. The three cards are a 48 w DC power supply card, a 23 w switching card (with an approximately 5 w core board on its back), and a 10 w amplifier card.

Because of the large size of the cards, there is a considerable power difference among the three different types. In addition, the power consumption of the components on an individual card is extremely unevenly distributed; it is low, and the slots relatively large, at the center of the cards, while power consumption is relatively large at the tops and bottoms of the cards. In terms of overall card layout, control cards (175 x 140 mm) are located between two rows of large cards. When using the conventional ventilation method, neither a long air path method with upward and downward flow nor a short air path with forward and backward air flow will meet requirements. A new method must be used so that each card in the frame and all components on the card will be in approximately equal ambient temperatures and so that the air flow rate will be minimized in order to assure a comfortable computer room environment. For this purpose we selected the design shown in Figure 7.

In this design, the air flows in through the bottom of the cabinet, enters the air intake space, then passes through the air passage and reaches the cards through round holes in the air passage; after the cold air flows over the components it passes out through the front of the card section, then exits into the computer room through slits around the door. Ventilation of the power supply drawers is the same as that used for the mainframe core storage.

Design of test cards. Resistors were installed on glass-reinforced plastic boards to simulate heat generation by the components. Three large cards,

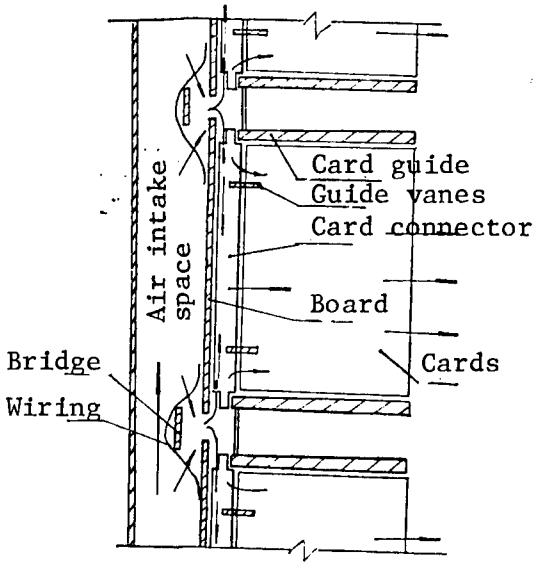


Figure 4. Short Air Path Ventilation, Type 3

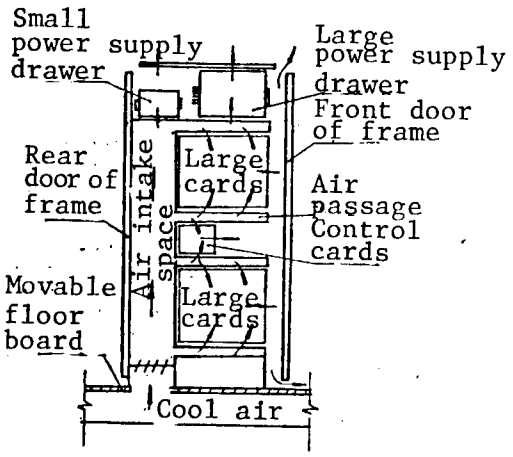


Figure 7. Ventilation of Peripheral Processor Core Storage Frame

the DC power supply card, the switching card, and the amplifier card, were installed in the test unit, spaced 28 mm apart. Figures 8 and 9 show the air flow results for the test unit with two different air flow designs. Either holes are uniformly spaced along the air passages, providing upward and downward ventilation of the cards (Figure 8), or holes can be spaced uniformly along the back edges of the card area at positions matching the boards to simulate flow through slits on both sides of the card connectors (Figure 9). When holes were drilled in the air passages, the real air speed was used. Because the cards are relatively tall, this type of air flow produces relatively serious problems in the air speed field along the cards.

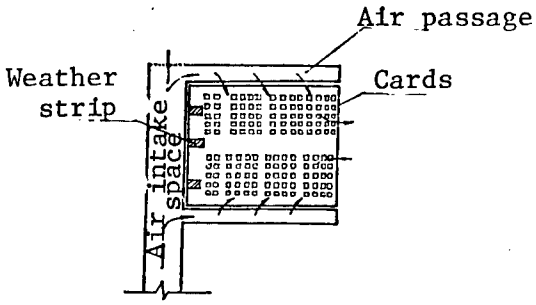


Figure 8. Experimental Unit for Ventilation by Top and Bottom Holes

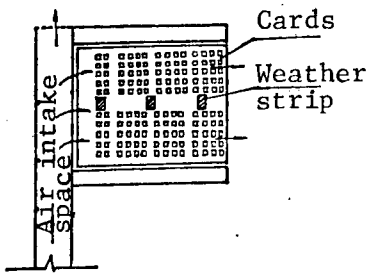


Figure 9. Experimental Unit for Ventilation With Holes in Back Plate

In order to assure that cards with different power consumption levels are in essentially the same ambient temperatures, the number of holes provided for each card is proportional to the card's power consumption level. Fifty-six round holes 4 mm in diameter were provided for the DC power supply cards,

40 holes 4 mm in diameter for the switching and amplifier cards, 10 holes 4 mm in diameter for the card boards, and 30 holes 4 mm in diameter for the 5 control cards.

For each card, the entering air first cooled the high-power consumption components at the tops and bottoms of the card, then cooled the low-power-consumption components in the middle of the card, so that each had a uniform temperature distribution.

In order to assure a uniform air speed through the small holes in the air passage, the principle of uniform air feed was used: because the main passages and side passages (i.e., the air intake space and air passages in this ventilation method) were of uniform cross section, as the air flowed through them the flow speed gradually decreased, so that the dynamic pressure gradually fell. If the local resistance and lengthwise resistance along the paths were relatively small, and were insufficient to compensate the dynamic pressure drop, then we find from the Bernoulli equation that the static pressure at the ends of the passages was higher than that at the beginnings of the passages, and accordingly the speed of the air exiting from the holes at the end was greater than that of the air exiting through the holes at the beginning of the passage. In order to equalize the speed of the air at the two ends of the passages, the ratio of the dynamic pressure to the total pressure must be decreased, i.e., the cross sectional area of the passage must be increased or the sizes of the holes decreased so that the exiting air speeds will be the same at all locations. But the dimensions are everywhere subject to the constraints of rational design and overall machine design, and accordingly the cross sectional dimensions must be calculated during design process and suitable values chosen.

Measurement Results

(1) Measurement Results From the Experimental Unit

(i) With the experimental model shown in Figure 8, holes were uniformly spaced in the air passage panels at the bottoms and tops of the cards; and the situations with and without baffles at the back of the card were compared. The baffles at the backs of the cards were used to prevent air short circuits and to make the temperature rise over the cards uniform. Figure 10(b) and Figure 11 show the ambient temperature on the cards and the temperature increase between the air entrance and exit.

(ii) Using the model shown in Figure 9, we spaced holes uniformly in the board at the back of the card area and compared the results obtained with and without baffles at the middle of the cards. Figure 10(a) shows the way in which the ambient temperature increases and the differences between the entrance and exit air temperatures are distributed over the DC power supply card. Providing baffles at the centers of the cards prevents air short circuits, so that the ambient temperature is fairly uniform over the cards. But because the power consumption at the tops and bottoms of the cards is high, while that in the middle is low, even with baffles added the temperature increases did not tend to become uniform.

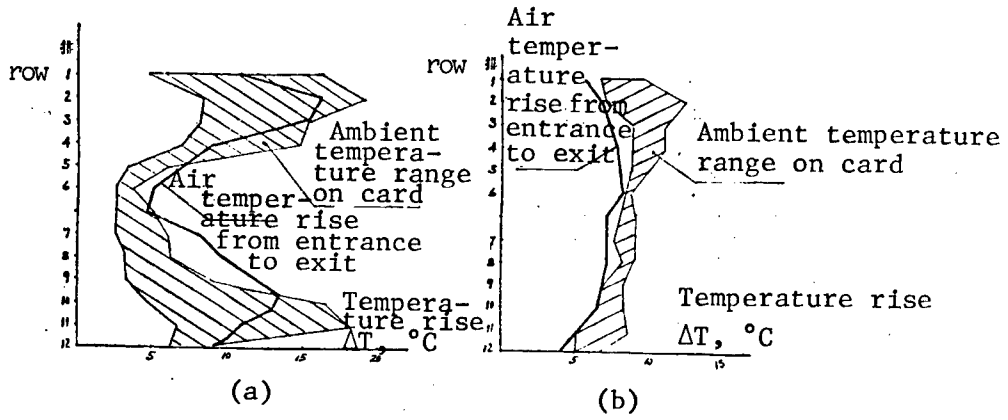


Figure 10. Temperature Rise on Cards, With Constant Air Source Flow Rate

(a) Uniform hole arrangement in board;
(b) Uniform hole arrangement above and below card.

Experimental conditions: card power consumption 48 w; 56 round holes 4 mm in diameter; static pressure in air intake space 3 mm H_2O; baffles added.

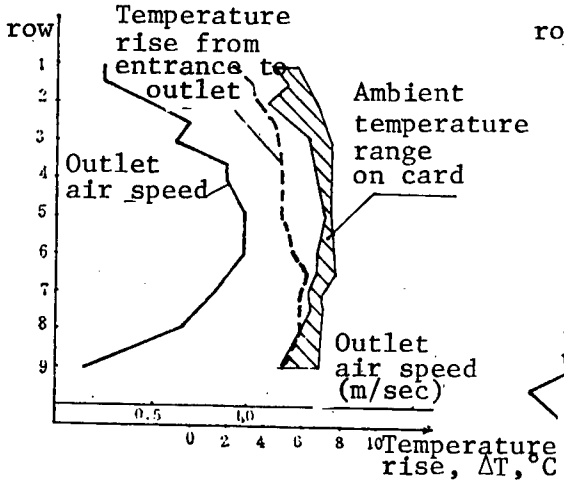


Figure 11. Temperature Rise on Experimental Switching and Amplifier Cards

Experimental conditions: card power consumption 23 + 10 = 33 w; 40 round ventilation holes 4 mm in diameter; static pressure in air intake space 3 mm H_2O; ventilation method of Figure 8 used.

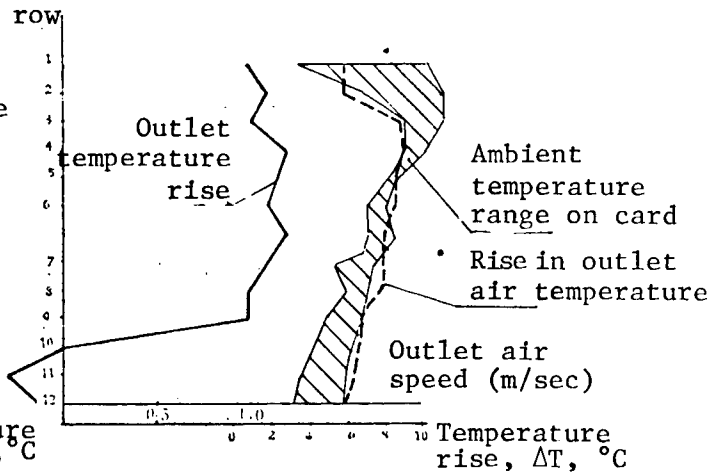


Figure 12. Temperature Rise on Actual Cards With Constant Air Flow

Experimental conditions: card power consumption 48 w; 56 air holes 4 mm in diameter; static pressure in air intake space 3.1 mm H_2O

(2) Test Results With Actual Frames

Figure 12 shows the distributions of the ambient temperature increase, the temperature rise from air inlet to air outlet, and the air outlet speed. This ventilation method was tested on actual frames and gave excellent results, with a uniform ambient temperature within the frame. Because the required ventilation air volume per watt of power consumption was only 0.3 m^3/hour, excellent heat dissipation results could be achieved with a low air flow rate. The low air flow rate means that the exit air speed is low and the exit air temperature rather high, so that service personnel in front of the frames will not feel cold. The design is simple; it is only necessary to control the number and distribution of the air holes and to choose suitable dimensions for the air passages and air intake space in order to effectively increase the uniformity of the temperature distribution.

The tendency for a uniform temperature distribution to develop over the large cards is also beneficial in this ventilation design because of the layout of the components on the boards. Therefore, when laying out the components, insofar as possible without degrading the interconnection capabilities, the air should be made to pass first over the high-power components, then over the low-power components; this will make the ambient temperature field on each board uniform. When the boards are in use, air short circuits should be prevented; an effective method is to place baffles on the boards. In addition, leakage from the air passages and air intake space should be prevented.

Use of baffles and prevention of leakage enable the cool air to be used with maximum efficiency and can increase the average outlet air temperature and produce comfortable conditions in the computer room.

With specific power consumption levels and air flow rates, we compared a design with air holes in the top of the air passage and a design with air holes in the back of the board (using 48 w DC power supply boards in all cases, with an air intake space static pressure of 3 mm H_2O, 56 round holes 4 mm in diameter per card, and baffles). The results are shown in Table 2.

Table 2. Comparison of Ventilation Techniques With Holes in Different Locations

Temperature rise		With air holes in air passage walls		With holes in back plate (actual cards)
		Experimental cards	Actual cards	
Ambient temperature rise on card (°C)	max	11.9	11.2	19.2
	min	5.2	3.2	2.4
	diff	6.7	8.0	16.8
Temperature rise from entrance to exit of frame (°C)	max	8.0	9.3	16.4
	min	3.8	5.5	3.8
	diff	4.2	3.8	12.6

Ambient temperature rise computations for the real boards indicated that the maximum temperature over a 375 mW triode would be approximately 58°C.

3. Thermal Design of the Mainframe Control Unit System

Because printed circuit boards are used in the frames for the mainframe unit's three control units, slots cannot be placed on both sides of the card connectors, and accordingly the ventilation scheme of Figure 2 cannot be used. In addition, because power cables enter through the guide areas, there is some inconvenience in using the ventilation scheme of Figure 3. As a result, we used the scheme shown in Figure 4. The fact that the printed circuit boards in the control units' frames are of floating type makes this ventilation method possible.

The system containing the three control units is in the form of a 12-sided polyhedron consisting of 11 frames and 1 door. Each frame has five small printed circuit boards, spaced 17 mm apart. The cooling air enters the air intake space in the 12-sided unit through a movable floor board, then flows through slots between pairs of small PC boards, then between pairs of card connectors, and enters the card area. After cooling the components, it exits from the front of the card section, then is exhausted into the computer room through slots around the doors of the frames (see Figure 13).

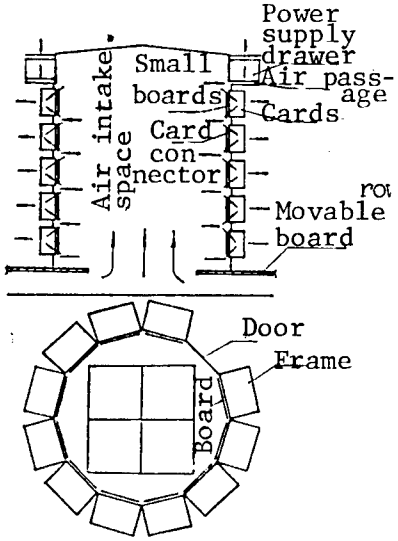


Figure 13. Ventilation of Three-Controller System of Mainframe

Simulation Experiment: In simulating the card, resistors were used to generate heat. Eighty were installed on each card. Figure 14 shows the way in which the temperature increases, and outlet air speeds are distributed over the card. It is evident from Figure 14(a) that the temperature distribution is highly nonuniform. When the cooling air flows around the card connector, because its flow speed is high, most of it flows along the slot between the two connectors, and when it reaches the center of the card it converges and flows out through the front of the card area; as a result,

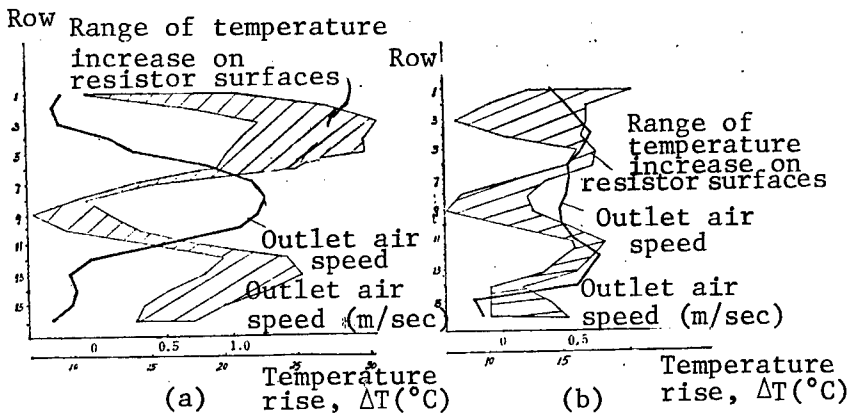


Figure 14. Outlet Air Speed and Temperature Rise in Three-Controller System Frame of Mainframe Unit

(a) Without air deflectors
(b) With air deflectors between connectors

Experimental conditions: static pressure in intake air space 2.8 mm H_2O (a) and 3.0 mm H_2O (b); card power consumption 15 w.

the flow speed is high when exiting from the central section of the card area, while it is low at the top and bottom, and suction develops. In rows 3-5 and 12-14 there are two high-temperature eddy areas. In order to make part of the air turn toward the front earlier, guide vanes were mounted between the two card connectors (shown in Figure 4), which decreased the temperature rise in the high-temperature eddy area so that the temperature distribution became relatively uniform over the entire card. Figure 14(b) shows the distribution of the temperature rise and air outlet air speed when the deflectors were added. It is evident from a comparison of Figures 14(a) and 14(b) that the deflectors provide a much more uniform exit air speed distribution, the suction in the upper and lower areas is alleviated and the temperature distribution becomes relatively uniform. Figure 15 shows the temperature rise for a 24 w test card.

Figure 16 shows the distribution of the temperature rise and exit air speed for an actual 15 w card. Figure 17 shows the effect of the signal cable cover on the temperature rise. It is evident that running the cable between the two small boards increases the air flow resistance, and that if the static pressure in the air intake space is not changed, too little air flows between the cards, and both the ambient temperature between the cards and the outlet air temperature rise somewhat, about 2-3°C. If the cables running between the two small boards are not pulled directly through, but instead are "bridged over" (see Figure 4), the air flow resistance is decreased and the temperature rise produced by the cable is relatively small, about 1°C.

As shown in Figure 16, calculations based on the ambient air temperature between cards and the thermal resistance of the IC packages indicate that the maximum junction temperature in a 375 mW high-speed shift register package is 63.5°C.

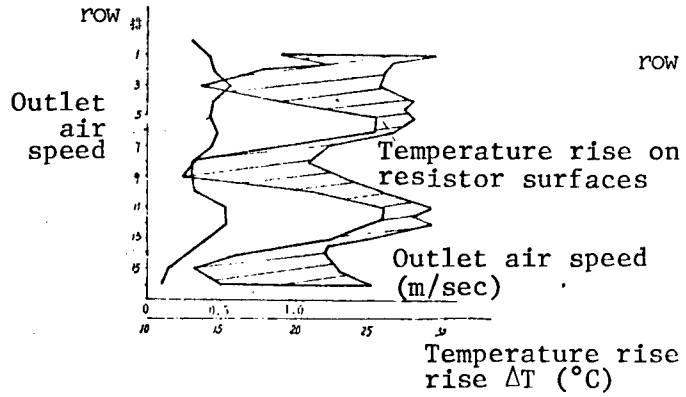


Figure 15. Outlet Air Speed and Temperature Rise for Test Cards of Mainframe Three-Controller System

Experimental conditions: Static pressure in air intake space 2.8 mm H_2O; card power consumption 24 w; deflectors between connectors; no power lines between small boards.

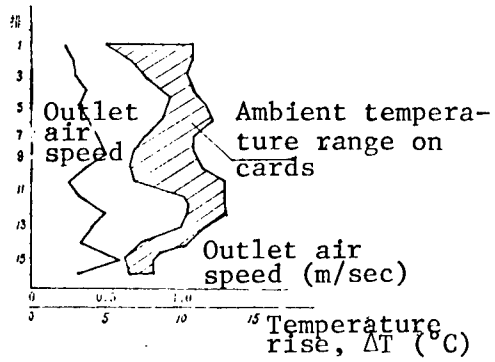


Figure 16. Outlet Air Speed and Temperature Rise on Actual Cards of Mainframe Three-Control Units

Experimental conditions: Static pressure in air intake space 4.6 mm H_2O; card power consumption 15 w; deflectors between connectors; 100 twisted-pair wires 0.6 mm in diameter run between small boards; no wiring bridge.

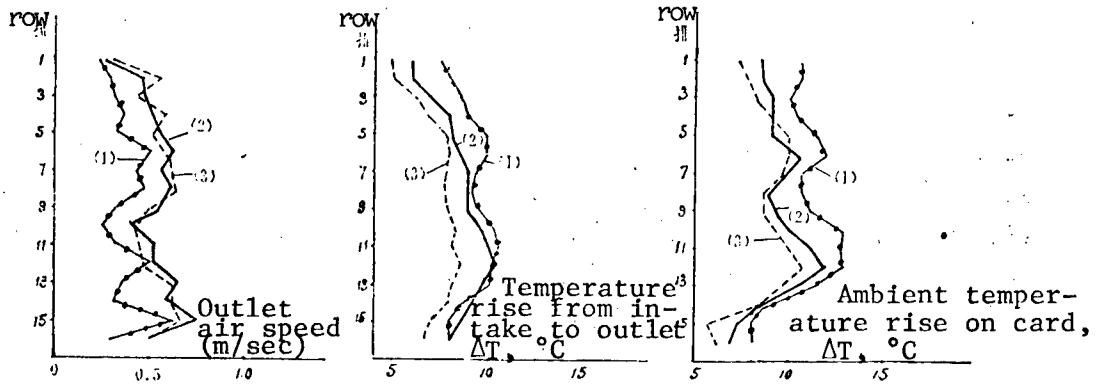


Figure 17. Effect of Wiring Between Small Boards on Temperature Rise and Outlet Air Speed in Three-Controller Frame of Mainframe Unit

Static pressure in intake air space 4.6 mm H_2O; card power consumption at 15 w. Curve 1, 100 twisted pairs 0.6 mm in diameter between small boards, no wiring bridge; Curve 2, 200 twisted pairs 0.6 mm in diameter between small boards, with wiring bridge; Curve 3, no wiring between small boards.

The ventilation method for the power supply drawers is the same as that used for the mainframe core storage power supply drawers.

The principal matters to which attention must be directed when using this ventilation method are as follows:

(1) The deflector dimensions and positions. The dimensions of the guide vanes and their positions and angles have a major effect on the air flow distribution. The dimensions and installation positions obtained in the experiments must be strictly followed.

(2) The air intake slot between the two small boards must not be blocked. If there is a relatively large number of connecting wires, either they must be "bridged across" or the space between the two small boards must be increased in order to assure that there will be sufficient air flow through the slot.

(3) Prevention of air leakage. The cold air which enters through the air space from the movable floor board must be permitted to pass only through the card area and must not short-circuit through other areas.

The air flow rates in the three control unit system of the mainframe are as follows: assuming a power consumption of 15 w per card, the rate should be 0.3 m^3/hour for each watt. The three-control unit system of the mainframe has a total of 11 frames, each of which has 120 cards, and the total air flow for the cards is 6,050 m^3/hr. Together with the ventilation air for the power supply drawers, the total air flow rate for the control unit system of the mainframe is 8,250 m^3/hr; the static pressure in the air intake space is 6 mm H_2O.

Advantages of This Ventilation Method

(1) It is usable with printed circuit boards and with frames in which power cables enter through the guide area.

(2) It is simple in design; the only structural parts added for ventilation are the deflectors, the bridge, and the baffles in the small air passage for ventilation of the power supply. It is easy to install.

(3) With correct selection of the dimensions of the air intake space and the slots between the two boards, as well as of the positions and dimensions of the deflectors, uniform ventilation can be achieved and the temperature rise within the frame will be uniform. In addition, the problem of heat dissipation on relatively high-power cards (e.g., a power consumption of 24 w per card, with dimensions of 175 x 278 mm) can be solved with a relatively low air flow rate. Because the air flow rate per watt of power consumption is small (about 0.3 m^3/hour), the exit air temperature is relatively high and the area in front of the frames will not be cold, so that conditions in the computer room will be comfortable. If the front doors of the frames are open during adjustment of the computer, the ventilation results will not be degraded.

Disadvantages

(1) The amount of air fed to each card is about the same; the air flow rate cannot be varied in accordance with the power consumption of each card.

Therefore, the temperature increases are different on cards with different power consumption levels.

(2) The deflectors must be installed precisely, because their locations and dimensions have a major effect on air flow; in addition, care must be taken that wires passing between the two small boards do not block the air intake space.

4. Future Utilization of the Short-Path Uniform Ventilation Method

The short path uniform ventilation method may be expected to be used in frames with layout densities greater than that of the 757 large-scale machine. If the ventilation scheme of the control unit frames is used, constraints will result from the use of printed circuit boards and installation of the power cables in the guide spaces. In addition, provided that the air intake space is sufficiently large, the sizes of the air intake slots or ventilation holes can be increased as the power consumption of the IC packages is increased. Thus, without an excessively high air pressure, an increase in the air flow rate and the air speed over the components can be used to cool rather high-power cards. The control unit system of the mainframe consists of a polyhedron 2 m in diameter inside which is a large air intake space; this design decreases the length of the wiring used and creates excellent conditions for ventilation and heat dissipation.

8480/9365
CSO: 4008/249

DESIGN OF ARITHMETIC COMPONENTS OF 757 VECTOR PROCESSOR PIPELINE

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 3, 1984 pp 21-28

[Article by Xu Jun [1776 3182] and Xu Kunming [1776 2492 2494], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] I. Design Requirements

The 757 machine is the first pipelined vector processor designed in China; it uses a pipeline for all vector and scalar computations. As a result, a high arithmetic speed had to be achieved, and in addition to vectoring the computation problems, the single-flow pipeline speed had to be made rather fast. Furthermore, a great variety of vector pipeline operations had to be added to the vector pipeline, which posed stringent requirements regarding the design of the ALU [arithmetic-logic unit].

In order to design the 757's ALU as a technically advanced, high-speed, reliable device with economy of materials, we focused attention on the following points during the design process:

1. The arithmetic components can perform a total of 76 instructions, including 70 vector operations and 70 scalar instructions (not all instructions are usable for vector and scalar computations). Vector computations were made primary, with additional provision for scalar computations, and the speed of scalar computation instructions was maximized.

2. Because of constraints imposed by the current level of development of Chinese-made components, using all Chinese-made small-scale integration [SSI] circuitry (and a small amount of medium-scale integration [MSI] circuitry) to design the arithmetic components of a super high-speed large computer was technically rather difficult; the logic chain for each beat contained a rather considerable delay, which, together with the unavoidable wait time for entry of voltages into flip-flops, the load allocation serial gate time, and the clock pulse alignment deviation, made the number of logic levels actually usable in logic small. As a result, our only recourse was to focus on arithmetic techniques, logic forms, saving hardware and decreasing the number of levels in the logic chain.

3. We studied methods of expressing the numbers used in the computations, the choice of a coding system, and the roundoff method in ordinary [noncomplemented] code to see which choices would increase convenience and would minimize the chance of unnecessary computation errors resulting from rounding off.

4. In order to increase the reliability of the ALU, we designed a parity prediction-parity test system within the ALU, which was combined into an organic whole with recomputation and a diagnosis and location system, and made full use of RAS [reliability, availability, and serviceability] techniques.

II. Operating Principles of the Arithmetic Components

When the instruction control unit's (ZK) instruction buffer register discovers that the instruction fetched from the instruction buffer store is an arithmetic instruction, it immediately issues a fetch-store queue request to the memory control unit (CK) at the same time sending the data string length and the execution starting address in storage which has been determined by the instruction control unit's address adder; in addition, it sends the instruction to the operation control unit's (YK) instruction buffer station and changes the internal data store fetch or store address in it to the code given by the vector look-ahead data fetch buffer register (X_{0_3}) or the vector look-behind data store buffer register (H_{0_1}).

When the operation control unit (YK) receives the pseudocode from the instruction control unit (ZK), it passes through two buffer levels, fetching the pseudoinstruction which is stored in the buffer station to the pseudoinstruction buffer register and fetches two operands from the vector look-ahead fetch register or vector accumulator register (L_{0_11}) or other registers as specified by the contents of the instruction, then sends them to the ALU's operand receiving registers M and N; at the same time, it enters into the ALU's operation command receiving register Q the opcode Q, the vector component serial number Fg, the vector/scalar flag T_B, the arithmetic control value K, the address D of the vector look-behind store register or vector accumulator register which stores the result, and other relevant operational information such as the byte address, byte length, half-word floating point numbers and the like.

In the case of a vector instruction, this activity may be repeated as many as 16 times or as few as once; the number of beats separating the repetitions depends on the pipeline timing for each operation, which is determined in all cases by the "enable current" or "enable new" signal sent out by the ALU. The "enable current" signal enables only the components of the current vector instruction to be pipelined, while the "enable new" signal enables different instructions and different operations to be pipelined. If the operand for the new instruction has not yet arrived, even if the ALU has issued the "enable current" and "enable new" signals the operation control unit cannot send an operation command to the ALU, but must wait until the operand arrives; this is commonly known as a "dependent operation" and its flowchart is shown in Figure 1.

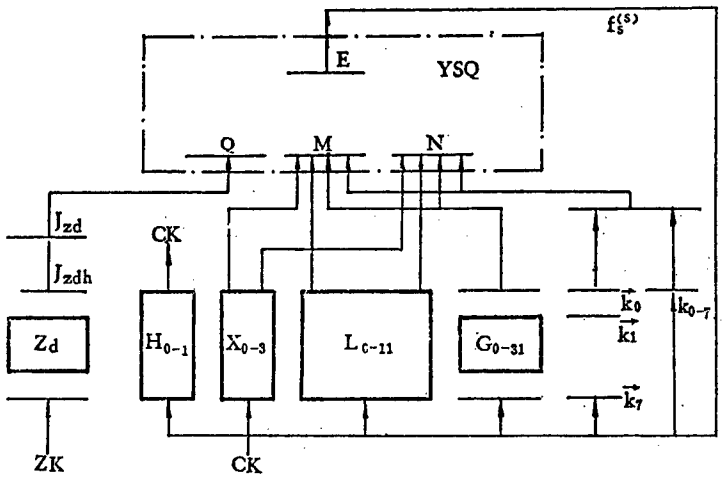


Figure 1. Block Diagram of ALU and Operation Control Unit

Accordingly, the ALU's flow of instructions actually can begin only after the instruction and the operand have both reached the ALU's receiving registers. The full pipeline in the ALU is divided into five stations; this is the longest pipe length, but not all instruction must pass through all five stations. More than 80 percent of the instructions can jump from station 1 to station 4, then pass through station 5. When a dependent operation arises, the result in station 4 can be shunted directly to M or N (i.e., f_s^4) in order to decrease the waiting time involved. For correctness, the arithmetic result still must be sent through station 5 in the normal manner so as not to affect the contents of the registers, as shown in Figure 2. Jumping of stations and shunting are both intended to maximize the performance of both scalar operations and dependent operations in the ALU. The basic capabilities of the pipeline stations are as follows:

1. Basic Functions of Station 1: 1) In floating point addition or subtraction, it finds the exponent difference from Q_i and sends the larger exponent to A_j, the corresponding mantissa to the SBS register, and the mantissa of the smaller number to register B_s, the alignment number is sent to register A_s. 2) In multiplication, the partial multiplier (27 or 29 bits) is sent to A_s, the multiplicand is sent to B_s, and $\frac{3}{4}(B_s)$ is sent to the SBS register. 3) In division, the initial iteration number K for the divisor found from division table T_x is sent to A_s.

2. In Functions of Station 2. 1) It determines the partial products with carries eliminated. In the multiplier decoder YM_{0-9}, the control voltages (KB, $K\frac{1}{2}B$, $K\frac{1}{4}B$, $K\frac{3}{4}B$, KFB) decoded for groups of 3 bits. The multiple gate BM that controls B_s selects nine decoded multiples and enters them in the truncated carry adder JFS, then finally finds sum H and carry number J. 2) It performs single-word or double-word alignment. The alignment uses a 1-beat method in which the fractional part can be shifted rightward by 0 to 63 places.

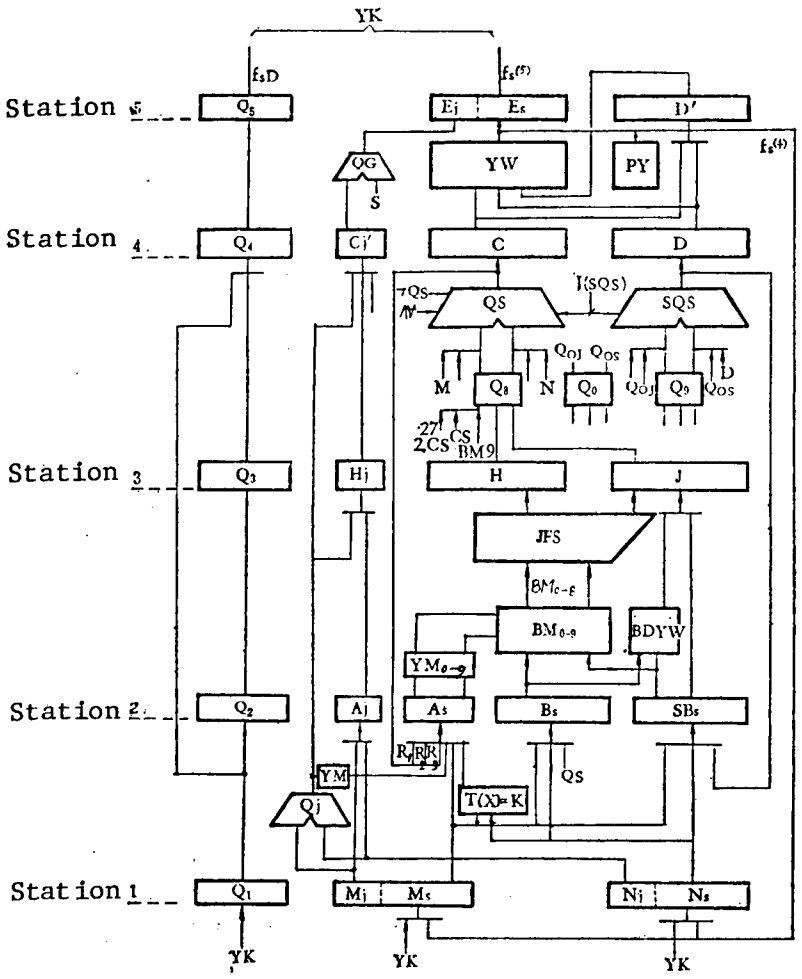
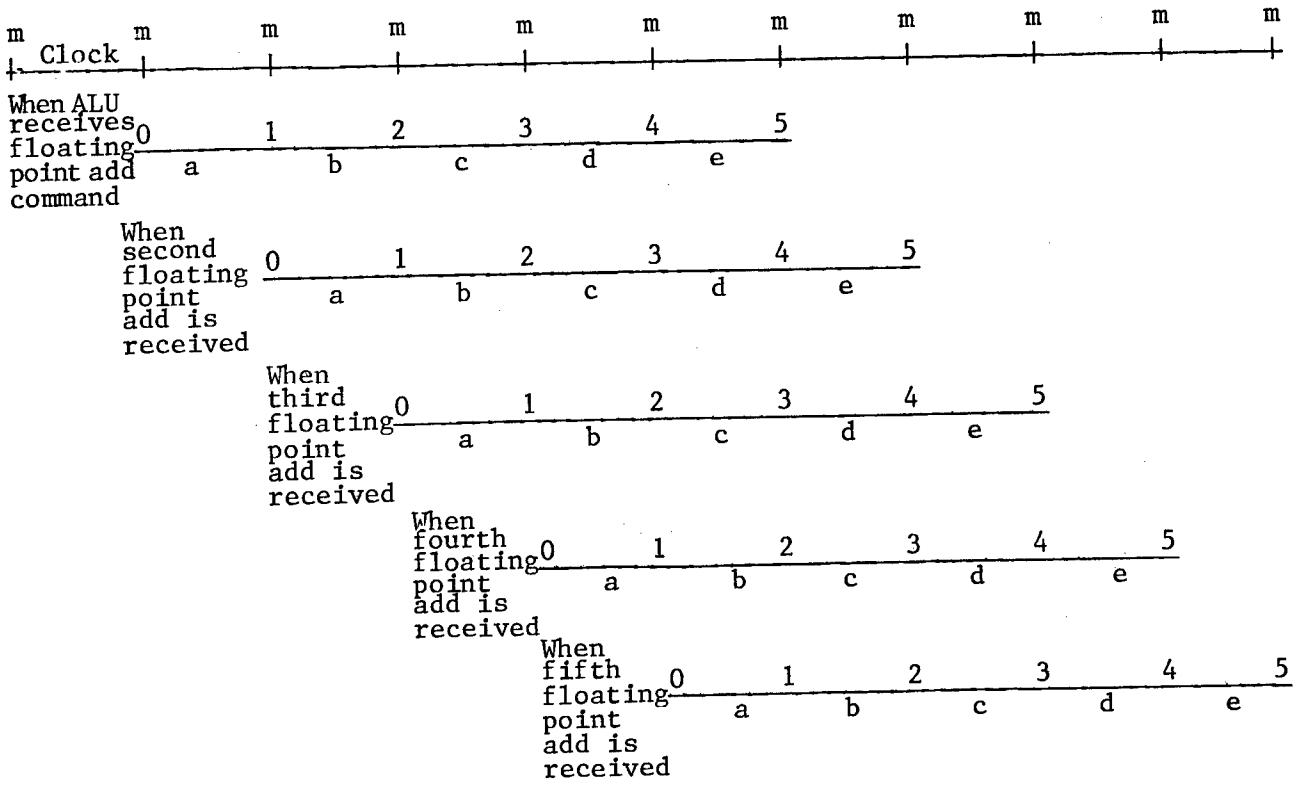


Figure 2.

3. Functions of Station 3. 1) It finds the partial and complete products from Q_S, as well as the intermediate products in iterative division, sums, doubled values, and high-order bits of numbers, and performs complementation and radix-minus-one complementation and other logical operations. 2) It obtains three-fourths of the multiplicand from SQS, as well as triple intermediate results in division, the low-order part of the full product, doubled values, and low-order bits, as well as performing other operations.

4. Functions of Station 4. 1) It uses a shift circuit (YW) to shift and normalize single and double words (1 beat for single words, 2 beats for double words). 2) It uses a voltage combining circuit (PY) or shift circuit (YW) to combine half-word floating point numbers of 8-, 16-, or 32-bit integers into full-word-length numbers in the manner specified.

5. Functions of Station 5. 1) It tests computation results. The parity test bit of register E is used to test whether the results are correct, and if incorrect, a jump is made to the recompute routine. 2) It transmits correct



Key: a. Find exponent difference (Station 1)
b. Alignment (Station 2)
c. Add and round off (Station 3)
d. Normalize (Station 4)
e. Transmit result (Station 5)

Figure 3. Overlap of Component Operations in Pipeline for Floating Point Add Vector Instructions

results. In order to increase reliability, the data code in E is formed into a 72-bit odd-weighted parity code for transmission, and the parity check of the data code sent from L or another register is made at the entrance to the ALU [Chinese abbreviation YSQ], if an error is found, the diagnostic interrupt routine can be used to make a soft correction of the odd-weighted parity code; thus a closed circuit redundancy is formed between the ALU and its peripheral registers, which effectively increases reliability. In particular, accumulator registers cannot be recomputed, but they can be corrected by means of the soft correction. 3) It transmits computation result time buffers.

As stated above, the results can be transmitted from f_s^4 of station 4, but there is insufficient time to transmit them from there to the accumulator or similar locations, since this cannot be completed in a single beat. This is why this station was added. Adding a station is not equivalent to adding time to the pipeline. For example, in scalar dependency, f_s^4 sends the results to M and N without loss of time, but in the case of a vector instruction, 16 component operations must flow through in a single instruction, and even at the

greatest flow speed, the next instruction still can start only after 16 beats; at this time the first component will long since have been completed, so that no loss of time will result from adding the fifth station.

Figures 3, 4, and 5 show the execution of vector operations in add, multiply, and divide instructions. It can be seen that all operations are executed with a high degree of parallelism and overlap, and all operations share all features of YSQ [ALU] to a rational degree, so that they can all operate in orderly fashion under pipeline conditions. The pipeline timing of each instruction must be carefully arranged, since either too fast or too slow a timing can produce collisions.

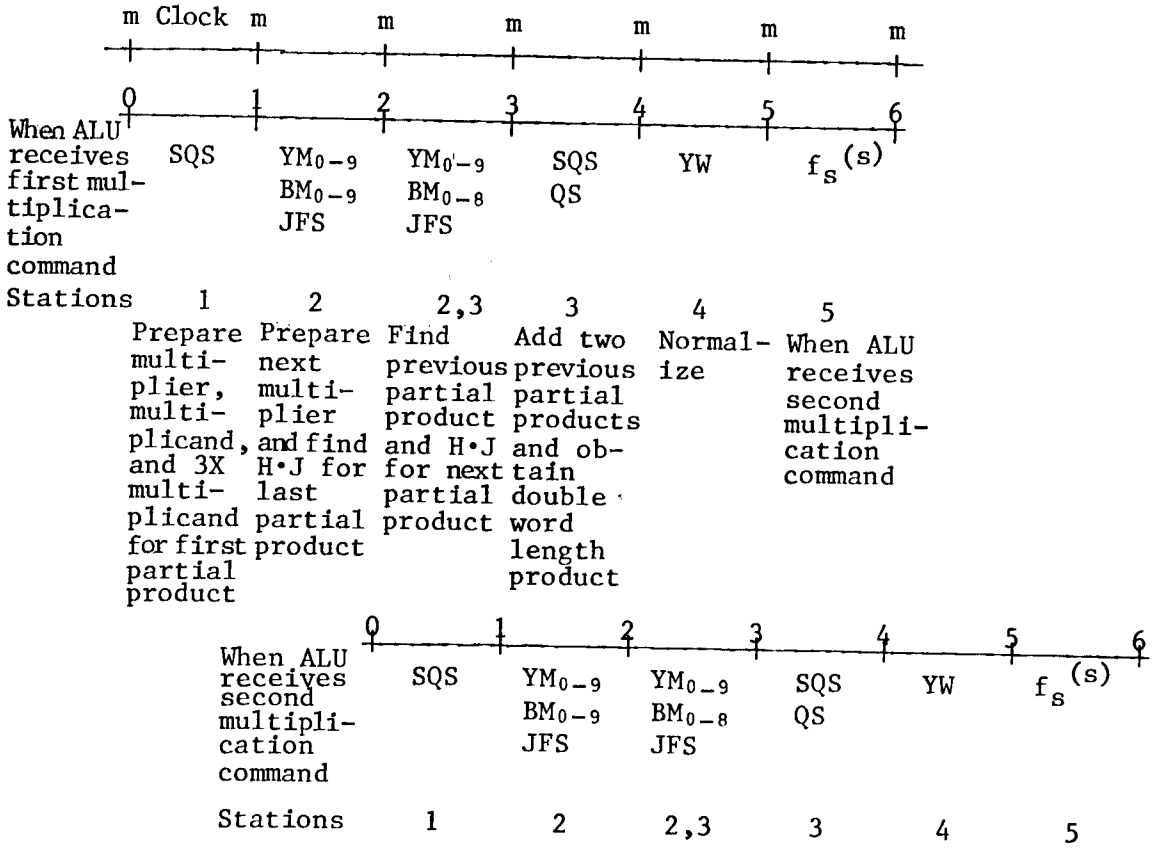
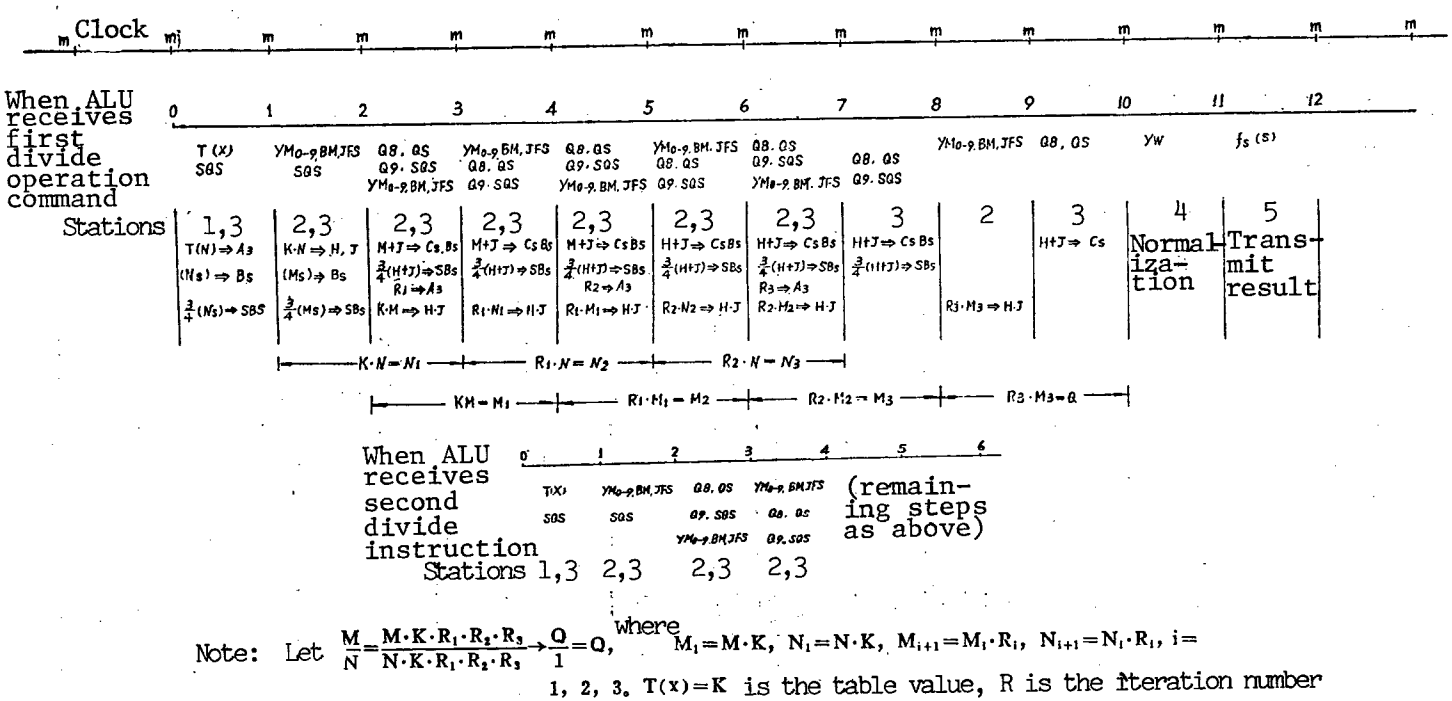


Figure 4. Overlap of Component Operations in Pipeline for Floating Point Multiply Vector Instruction



Note: Let $\frac{M}{N}=\frac{M\cdot K\cdot R_1\cdot R_2\cdot R_3}{N\cdot K\cdot R_1\cdot R_2\cdot R_3}\rightarrow\frac{Q}{1}=Q$, where $M_1=M\cdot K$, $N_1=N\cdot K$, $M_{i+1}=M_i\cdot R_i$, $N_{i+1}=N_i\cdot R_i$, i= 1, 2, 3. T(x)=K is the table value, R is the iteration number

Figure 5. Overlap in Pipeline When Executing Component Operations of Floating Point Division Vector Instruction

III. Number Representation, Choice of Coding Systems, and Ordinary Code Roundoff Rules

A computer's word length is determined by the type of problems to be processed, the storage capacity and the computing speed. When the 757 machine's word length was fixed at 64 bits, the topic with a major bearing in ALU logic design was the representation of full word length floating point numbers, half word length floating point numbers, and double word length floating point numbers. These three types are represented as follows:

1. Full word length floating point numbers

$$SF = 2^p \cdot s, \quad -128 \leqslant p < 127, \quad -(1-2^{-55}) \leqslant s \leqslant 1-2^{-55}$$

2. Half word length floating point numbers

$$SF_{1/2} = 2^p \cdot s, \quad -128 \leqslant p < 127, \quad -(1-2^{-23}) \leqslant s \leqslant 1-2^{-23}$$

3. Double word length floating point numbers

$$SF_2 = 2^p \cdot s, \quad -128 \leqslant p \leqslant 127, \quad -(1-2^{-110}) \leqslant s \leqslant 1-2^{-110}$$

In these representations, the exponent is represented in base-2 complement form and the mantissa in ordinary [uncomplemented] code.

The choice of base was made primarily in order to achieve maximum precision in the representation of half-word floating point numbers. Using base 2 or base 16 has its advantages and disadvantages, but based on considerations of implementation and familiarity, we continued to use the representation employed in the Model 109 (Mod 4) which we developed.

Very few high-speed computers represent data in ordinary code because when it is used for computations, if the difference is negative an additional beat of code conversion time is required, and when rounding off the ordinary code a relatively long end-around carry is necessary; as a result, all Chinese-made large computers use complement arithmetic. In the logic design of the 757 machine's ALU, because addition, roundoff, and code conversion can be carried out in a single beat of adder time, we changed over to the ordinary code representation (see section on adder logic design). Use of ordinary code has many advantages: it is relatively easy to understand, the absolute-value "round down at 4, round up at 5" method can be used, the error distribution is uniform, good symmetry is achieved, and relatively high computation accuracy is obtainable. In addition, normalization, shifting, and multiplication and division use simple logic and convenient operations and given an economy of hardware.

An example of rounding off:

$$\text{roundoff} = (C_{opp} + C_{\text{equal exp}}) \cdot J(Qs)1/55 + C_{\text{equal exp}}(C_+ \cdot H_{56} + C_- \cdot H_{56})$$

where C_{opp} indicates that the two numbers entering the adder have opposite sign; $J(Qs)1/55$ indicates bits 1-55 in adder Qs with a carry bit in the highest-order digit; and H_{56} indicates the value of the roundoff bit H_{56} of fractional part H.

Example of handling of coding system:

$$\text{Complement } Qs = \neg[\overline{C_{opp}} + C_{opp} \cdot J(Qs)1/55]$$

IV. Design Features

1. Full use is made of the ECL circuits' DOT-OR capability, with a clever logic design incorporating alternating positive and negative logic, thus saving hardware, decreasing the number of gate levels and saving level delay time.

2. With pipelining of vector and scalar operations, nondependent operations can be executed rapidly in the pipeline.

3. Because ordinary-code rounding off is used, the error distribution is uniform, with good symmetry and good computation accuracy.

4. The single-ALU design gives a high device utilization rate.

5. A special-design fast algorithm is used:

(5.1) Alignment in 1 beat, and 1-bit normalization ordinary-code roundoff floating point addition.

(5.2) Fixed point or floating point multiplication of 27-29 bits, 3 bits at a time in groups of 3.

(5.3) Precise iterative division with threefold iteration.

(5.4) Fast pipelined double-word floating point addition.

6. An end-around carry adder with ordinary code roundoff and a high-speed carry chain is used. Code conversions are all made in the adder, and accordingly there is no increase in the number of add beats, so that a full add uses only 7.5 levels. When this was engineered the actual time was 33-43 ns (including wiring delay; the gate delay time was 4 ns).

7. The ALU components have a rather complete and stringent test system.

(7.1) The parity prediction parity check testing method is primarily used, giving coverage of 95 percent or more. The device utilization ratio is rather low. The table below gives the component ratios in test systems that have been described in the literature.

Unit \ Model	013	905	757
Operation control unit	22%	18.1%	10%
Adder	50%	35%	<25%

(7.2) A closed protective Hamming code is formed outside the ALU, i.e., between the ALU and the accumulators, thus effectively assuring correct computation.

(7.3) Timely warning when a malfunction is detected, with recomputation, diagnosis, and location.

V. Execution Speed of Main Insturctions

Type	Name of instruction	Beats to execute each scalar component	Number of beats in pipeline for each component	Remarks
Move	W ⇒, L → Y, Y → L, QCK h → L, K → h, L → G, L → BH G ⇒ D, D → L, O → L, L ↑ k → ko L → q	3	1	
Comparison	Find maximum	Scalar 6, vector 2L' + 6	2	
	Find minimum	Scalar 6, vector 2L' + 6	2	
	Maximum modulo x	Scalar 6, vector 2L' + 6	2	
	Minimum modulo x	Scalar 6, vector 2L' + 6	2	
	Larger than vector	Vector L' + 5	1	Vector operation
	Smaller than vector	Vector L' + 5	1	" "
	Equal to vector	Vector L' + 5	1	" "
	Smaller than comparison	3	1	
	Larger than comparison	3	1	
	Equal to comparison	3	1	
	Smaller than, modulo x	3	1	
	Larger than, modulo x	3	1	
Logical operations	K logical add	4	2	Scalar only
	K logical multiply	4	2	" "
	K bitwise add	4	2	" "
	Logical add	3	1	
	Logical multiply	3	1	
	Bitwise add	3	1	
	Complement	3	1	
Exponent code operations	Exponent add	3	1	
	Exponent subtract	3	1	
	Exponent code add immediate	3	1	
	Exponent code subtract immediate	3	1	

[continued]

[Continuation of Execution Speed of Main Instructions]

Type	Name of instruction	Beats to execute each scalar component	Number of beats in pipeline for each component	Remarks
	Exponent special shift	3	1	
	Exponent change to floating point	3	1	
Addition	Add	5	1	
	Subtract	5	1	
	Complement subtract	5	1	
	No-normalize no-roundoff add	5	1	
	No-normalize no-roundoff subtract	5	1	
	Special add	L' + 4	1	Vector instruction
	Double precision add	7	6	
	Double precision subtract	7	6	
Integer add	Integer	3	1	
	Integer subtract	3	1	
	Integer iterative add	L' + 3	1	Vector instruction
Multiplication	Normalized rounded multiply	6	2	
	Self multiply	6	2	
	Integer multiply	6	2	
	No-normalize no-roundoff multiply	7	2	
Division	Divide	12	8	
	Complement divide	12	8	
Shift	Integer left shift	3	1	
	Integer right shift	3	1	
	Left shift immediate	3	1	
	Right shift immediate	3	1	
	Integer double word length left shift	4	2	
	Integer double word length right shift	4	2	
	Immediate double word length left shift	4	2	
	Immediate double word length right shift	4	2	

[continued]

[Continuation of Execution Speed of Main Instructions]

Type	Name of instruction	Beats to execute each scalar component	Number of beats in pipeline for each component	Remarks
K operations	Determine K characteristic	3	1	No vector operation
	K → L	3	1	" "
	L → K	3	1	" "
Insert	Integer integer insert			
	Half word floating point integer address insert			
	Contract, store integer or half-word floating point	L' + 2		Vector instruction
Other	Count 1	67		Serial
	Count left 0	max 67		Serial (3-67 beats)
	Change sign	3	1	
	Full word round off to half word	3	1	
	Separate integer, fraction	7	7	
	Separate integer to exponent	5	5	

Note: L' is vector length

VI. Conclusions

A long period of operation has indicated that the arithmetic components' main capabilities meet design requirements, and no major errors have shown up in the algorithms; one fairly major error has been the lack of special handling in division of zero by a nonzero number. A more efficient layout of the connections between the ALU and external components could make it possible to use higher frequencies. Because of the complexities attendant upon pipelined vector computations and insufficient experience in debugging of testing systems, there are still difficulties in debugging.

In terms of design there are two major deficiencies.

1. Little reserve potential. Because SSI circuitry was used to produce a high-speed pipelined computer, the wiring connections were too long and used up too much time. As a result, although the number of logic levels used up by each logic chain does not exceed the design requirements, time is still felt to be inadequate, which has hindered further increases in the working frequency. Two other factors also contribute to the long delay time in the wiring: one is the efficiency of the wiring layout, and the other is that the use of printed circuits in the computer increases the wiring delay time, so that in some locations the only way to decrease the time used was to make use of straight wires between points.

2. The high speed of the arithmetic components should be matched to the overall speed of the machine. Currently, the main factor affecting computation speed involves not the arithmetic components, but shifting, instruction fetch, and data fetch and store. Therefore, simply requiring that the scalar instruction execution speed of the ALU be increased will not have much of an effect on the machine as a whole. As soon as a scalar instruction appears, the ALU's waiting time becomes long. This is because a comprehensive view was not taken during overall machine design.

Above we have summarized certain design results involving the use of SSI circuitry to produce a high-speed pipelined computer, related to specific topics. In brief, the objective in designing the arithmetic components was to achieve a high level of capabilities and to make them into a rather technically advanced high-speed, reliable, practical ALU.

Other comrades who participated in the logic design of the 757's ALU included Comrades Wang Pixian [3769 0012 7341], Fu Chaoyuan [0102 2600 0337], Chen Dingxing [7115 1353 5281], Tian Gongxing [3944 0361 5281], Man Yunxia [3341 6663 7209], and Li Xiuying [2621 4423 5391], and subsequently Comrade Hou Qi [0186 0796].

8480/9365
CSO: 4008/249

USE OF 5YCZ-72S CARD CONNECTOR IN 757 COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 3, 1984 pp 46-50

[Article by Chen Lizhong [7115 4539 1813] and Wu Fangyuan [0702 2455 0337], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] The 757 computer is a large, fast general-purpose computer developed in its entirety with Chinese-made components and technologies. The machine contains a total of 12,000 connector sets, half of which are newly developed products such as the 5YCZ-72S card connector, the SJC-108 adaptor socket, and the like.

The 5YCZ-72S card connector, developed to meet the computer's assembly requirements, is designed for both plug-in and wire-wrap connection. Its development promoted the application and wider use of wire-wrap technology in computers. The 757 computer uses more than 5,000 of the connectors; their use in the machine has shown that they give good performance, have reliable quality, and meet machine requirements.

When this connector was developed, China had not previously developed one successfully. It had to meet not only general connector requirements, but also wire-wrap connection requirements. Accordingly, assuring its quality became one of the key problems in the development of the computer.

Below we describe the process of applying the 5YCZ-72S card connector in the 757 computer, focusing on user analyses of its capabilities and measurements of its quality both before and after it was incorporated into the computer and the requirements posed by its utilization.

I. Choice of a Design for the 5YCZ-72S and Main Technical Requirements

Because the 757 computer system is very large, has high performance standards, and uses a large number of connectors, and because the 5YCZ-72S card connector is used to connect cards to boards, the principal objective in its development was assuring its quality. Accordingly, we analyzed the design of widely used foreign and domestic connector springs and selected the "figure 9" design shown in Figure 1.

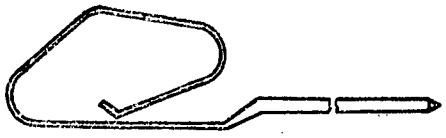


Figure 1. Shape of Spring Strip

This design uses material of variable cross section. Its main design dimensions are: wire-wrap tail of square cross section, 0.6 x 0.6 mm^2; contact spring thickness 0.2 mm, width 1.8 mm, with 0.4 mm slot in center. The main advantages of this design were its good load-bearing ability and ability to regain its shape after deformation, its long life and rather stable insertion and withdrawal force and contact resistance. But the forming process involved in producing it was rather complex and difficult to automate, and some difficulty was involved in assuring uniformity of shape.

The main technical requirements were as follows:

Spring Strip Section. Total withdrawal force, at least 4 kg; plug-in life, 1,000 times; contact resistance, maximum 10 mohm during rated lifetime, maximum 15 mohm after rated lifetime; height of contact point, at least 2 mm; contact stability: spring sections on both sides of central slot must be able to maintain contact with card leads; and surface plating, thickness 3-4 μm, in dense layer.

Wire-wrap Posts. Corner radius, minimum 0.04 mm, maximum 0.08 mm; hardness H_V, 160-225; flatness, maximum 0.05 mm in 10 mm; and burr, maximum 0.02 mm.

II. Uniformity of Spring Shape: The Key to Quality

The shape of the spring strips in the 5YCZ-72S is complex, the precision requirements are high, and the process is difficult to automate; this poses problems for uniform shaping of the springs. But the degree of uniformity affects not only the appearance of the product, but also its quality and contact reliability. In order to increase spring uniformity, the following main processes must be stringently controlled.

(1) Precision of square wire and variation of thickness in flattening. As shown in Figure 1, the spring strips are formed from square wire 0.6 mm on aside that has been rolled from flat wire, and a section of which is then flattened, after which it is cut and subjected to several bending and forming processes.

It is evident from Table 1 that the precision of the square wire and its thickness variation have a major effect on the width after flattening; excessive variation in thickness will create differences in curvature and resilience when it is bent, thus affecting uniformity of shape. Rather large variations in width also affect centering in the subsequent cutting

of the slot. Accordingly, the precision of the square wire and the thickness variation on flattening must be stringently controlled. We used width grading to compensate for the effect of width variations, assuring that the slot was centered.

Table 1. Effect of Deviations in Thickness of Square Wire and Flattened Section

Deviation of square wire	Thickness deviation of flat section	Width change on flattening
± 0.01 mm	0	0.12 mm
0	± 0.01 mm	0.18 mm
± 0.01 mm	± 0.01 mm	0.30 mm

(2) The hardness of the strips must meet requirements. In order to eliminate stresses and brittleness produced by the flattening process, heat treatment is required; if the conditions are suitably chosen, the hardness requirements for wire wrapping will be met along with the elasticity and shape requirements for the spring strip. In addition, the hardness uniformity of each lot of spring material has a direct effect on uniformity of spring shape.

(3) Strict quantitative sample measurements in the forming process. Stringent control of the processes described above can steadily improve the acceptance rate in forming the spring strips, but it cannot completely eliminate nonuniformity in their shape. Accordingly, it is also necessary to make quantitative sample measurements during the spring forming process; we used comparison of magnified projections as a means of random monitoring of changes in spring shape. When the shape was found to be outside the permissible limits, the causes were immediately determined and steps to eliminate them were taken.

(4) Guaranteeing uniform positioning of spring strips in the connector body. This means correct alignment within the body, uniform height, and a suitable top pressure state. This is very important for the stability of the insertion and withdrawal force and the contact resistance. Therefore, we designed an assembly jig, shown in Figure 2. First the positioning piece makes the heights of the spring pieces flush and trues them up laterally. Then the assembly fixture securely clamps the tail sections of the spring pieces; with the body of the housing held immobile, the jig is pulled so that all of the springs are simultaneously inserted into the body in suitable positions.

III. Product Testing and Analysis

With reference to the Ministry of the Electronics Industry's general technical requirements for connectors, we established the following routine testing requirements based on our applications needs: 1) external inspection; 2) measurements of main technical characteristics under normal conditions:

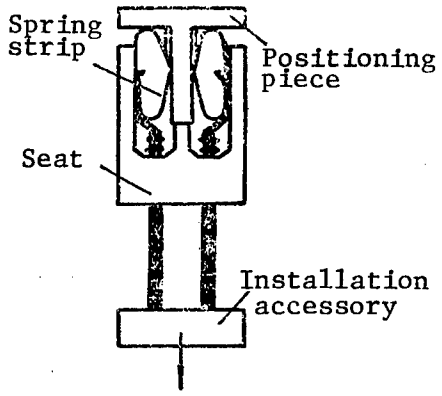


Figure 2. Installation of Spring Strips

ⓐ wire-wrap technology characteristics, ⓑ conventional connector technology characteristics, ⓒ accurate position of holes in body; 3) high and low temperature tests; 4) humidity tests; 5) vibration tests: ⓐ vibration strength, ⓑ vibration reliability; 6) insertion and removal tests; and 7) tests of plating quality.

The wire-wrap technology characteristics include the strength of the wire-wrap posts, cracks, cross sectional shape, corner radius, shape of tip, flatness, burrs, torsional rigidity, and hardness.

The principal plating quality characteristics tested are the thickness and compactness of the plated layer. The samples are immersed in nitric acid; the time and place of appearance of a green color are observed, and the integrity of the plated layer is determined. The thickness is determined with a beta-ray thickness measuring device.

If the tests reveal quality problems, they are studied and methods of correcting them are determined.

In addition, the tests revealed the problem of a decline in the insertion and withdrawal force. It was particularly rapid in connectors with a high insertion and withdrawal force. We devoted major attention to this problem: did it involve the elasticity of the spring strips themselves, or other factors, and what effect did it have on connector performance? We performed a large number of computations, experiments, and analyses, principally the following:

1) Measurement of the relationship of contact pressure to contact resistance. When the spring was in contact with a printed circuit board in the direction of loading, with a continuous increase in the pressure we found that the contact resistance changed in accordance with the curve shown in Figure 3. This curve was used to determine the pressure range which gave a stable contact.

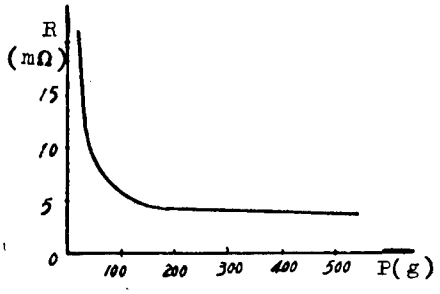


Figure 3. Relationship of Pressure (P) to Contact Resistance (R)

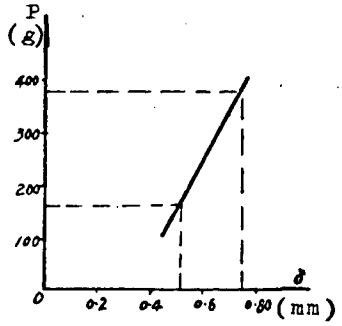


Figure 4. Relationship of Spring Deformation (δ) to Pressure (P)

It is apparent from Figure 3 that when the pressure exceeds 150 g the contact resistance becomes stable. The coefficient of friction on the plated surface is about 0.3, so that the withdrawal force for a single spring must be at least 45 g, and the total withdrawal force for an entire connector must be at least 4 kg. An excessive withdrawal force not only affects the spring's wear life, but also is unfavorable for its operation. According to our application conditions, while assuring reliable connections, it is desirable to minimize the total withdrawal force.

2) Calculation of spring loading and measurement of the relationship between spring deformation and applied pressure. Based on the spring's shape and dimensions, we calculated whether the pressure exceeded the elasticity limits of the spring material under normal conditions and whether it would be possible to assure a minimal contact resistance under the most unfavorable conditions. In addition, we measured the relationship of spring deformation to pressure, which is shown in Figure 4.

Under the conditions of actual use, the deformation of the spring was likely to be between 0.52 and 0.75 mm. It is evident from the curve that the pressure corresponding to this deformation range is 160-380 g. If the minimum pressure exceeds 150 g, a stable contact will be obtained. The calculated results agreed with measurements, indicating that the design was rational.

3) Monitoring of spring shape by projection. In order to monitor the reliability of the spring strips under maximum deformation, we removed eight strips from the ends and center of the connector body and used a projector to produce magnified images of their initial shape. They were then placed in the connector body and compressed until they were level with the spring seats. At specified time intervals the spring strips were removed and their projections were compared with the original projected shapes. After a year's measurements, the shapes of the springs showed no major changes, indicating that they had continuously remained within the permissible range of deformation.

4) Analysis of decay of withdrawal force. The above experiments and measurements showed that the decrease in the total withdrawal force did not result

from changes in the characteristics of the spring strips themselves; when it reached a certain range (not below the lower design limit), the decline in the withdrawal force leveled out. Accordingly we may assert that the decrease results from process factors and installation factors. In order to prove the correctness of the assertion, we tested connectors with a total withdrawal force of over 8 kg under the following conditions: I, careful selection of springs while removed from body, followed by reinstallation in the connector; II, removal of the springs, no selection, followed by reinstallation in the connector; III, springs not removed from original connector. The three test pieces were numbered I, II, and III accordingly. Insertion and withdrawal testing and storage gave the total withdrawal figures shown in Table 2.

Table 2. Reinstallation Comparison Experiment

Test piece	Remeasurement (kg)	Insertion-removal and storage tests (kg)							
		10 times	80 times	180 times	650 times	1,150 times	5 days	8 days	10 days
I	4	4.2	4	4	4	4.2	4	4	4
II	5	4.7	4.2	4.5	4.2	5	4.5	4.5	4.2
III	8	6.5	6	5.2	5	5.2	4.7	4.7	4.7

It is evident from the table that process factors and assembly factors directly affect the size of the initial withdrawal force and the extent of its decay.

5) Test of plug connection reliability and wire-wrap reliability. A general appraisal of comprehensive experiments on the wire-wrap performance of this product has already been made. In order to test the reliability of these two types of connections in use, we assembled a unit module equipped with 24 cards. The connection points and wire-wrap points of 48 connectors were connected in series by two paths. Each circuit had 1,344 plug connection points and wire-wrap connection points; a voltage was applied to them in a nonair-conditioned, contaminated environment. They were also subjected to impact and the cards were withdrawn and reinserted. At specified intervals the total resistance of each circuit was tested. After more than a year's monitoring, the total resistance of the circuits showed no significant change.

The above calculations, tests, and analyses enabled us to essentially master the main technical characteristics of the product and to meet design and use requirements so that it could be produced in lots for use in the computer.

IV. Comprehensive Testing Before Installation in the Computer

1) Choice of Tests. Even though the shaping and installation of the spring strips were subjected to rigorous control, we carried out comprehensive tests of their characteristics. In order to assure their absolute reliability, before installing them in the machine measurements also had to be taken on each connector. The measurements were as follows: ① correct height of contact points; ② contact of metal on both sides of slot with card leads.

We made a testing unit for testing each connector before it was installed, which gave excellent results.

2) Building the Test Unit

① Making the test head. The key element in the test unit is the test head. In order to test whether the metal on both sides of the slot is in contact with the card leads, we had to make a test head whose precision was greater than that of the cards. We divided each card lead into two leads separated by an 0.6 mm gap, resulting in a printed circuit test head with 144 lines. The production technology was the same as that for the printed circuit boards. The key was to select suitable matchup precision so as not to produce incorrect measurements. This test head was used to make tests on a 72-line card connector.

In order to screen out spring strips with a contact point below the specified height, the leads of the test head can be cut off at a certain height. For example, since we specified that the height of the contact point should be a minimum of 2 mm, we cut off the top 2 mm of the leads on the test head as shown in Figure 5. In this way, the springs with a contact point less than 2 mm high would not make contact with the test head.

② Testing method and circuit indication. The metal on the two sides of the slot in the spring strip was regarded as two series contacts corresponding to the two leads on the test head. The two connection points on each spring strip were connected in series with an indicator light by the leads on the test head, forming a circuit. Then the 72 spring circuits were connected in parallel, forming a circuit, as shown in Figure 6.

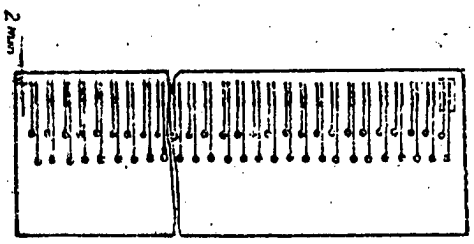


Figure 5. Test Head

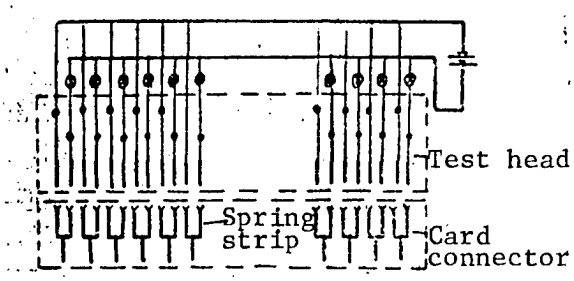


Figure 6. Test Circuit

When the test head is inserted into the connector, if both sides of the slot on the spring can contact the corresponding leads and the height of the contact points is at least the specified value, the corresponding indicator lamp lights and the spring passes the test. If the indicator lamp does not light, at least one side of the spring does not contact the test head or the contact location is too low, and this spring thus does not meet requirements. The number of the indicator lamp indicates which spring should be replaced.

This test unit was used to test all 72-line connectors before installation; the pass rate was high, and we were able to test 200-300 connectors an hour, with excellent results.

V. Use Requirements

Stringent quality measurements, tests, and analyses aimed at assuring reliable functioning of the connectors constitute one aspect of the problem; the other is their rational utilization and operation. These two aspects must both be taken into account in close association. Products which have been accepted but which are not used or operated properly are capable of causing product damage and malfunctions. There are many instances of this type. Accordingly, we established several requirements regarding the operation of the 72-line connectors.

(1) Stringent control of variation in the curvature and thickness of the cards during their production. The 757 computer's cards use a double 72-line insertion edge; they measure 278 mm (high) by 176 mm (wide) by 2.3 mm (thick) and consist of eight printed circuit layers as shown in Figure 7. With such large dimensions, there is a need for stringent control of their curvature and thickness tolerances during production. If these factors are too great, not only is the service life of the cards decreased, but there may be excessive variation in the contact pressures of the springs, which will affect their insertion and withdrawal and decrease connector service life. If the thickness is insufficient, poor connections may result. The design specifications are that along the greatest card dimension the degree of warping must not exceed 2 mm, while the thickness tolerance may not exceed ±0.1 mm.

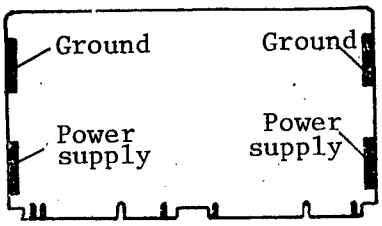


Figure 7. Card

The chamfer size of printed circuit cards affects the insertion force and spring wear. Because the contact points of the 72 springs are rather low, the chamfering of the cards must not be too great. The design requirement is that it be 0.5 mm x 45°. The cards must not have burrs or edges overlap.

(2) Special tools must be used for insertion and withdrawal. The 757 computer's boards consist of 5 board modules, each of which can hold 24 cards, as shown in Figure 8. The cards are inserted from the front along a horizontal guide. In the guide slots are spring strips serving as ground and power supply. Thus, special tools must be used for insertion and withdrawal to insure that the cards go in smoothly.

(3) Assuring centering of card and connector. It can be seen in Figure 8 that when the card begins to pass down the guide slot, it passes over the ground and power supply springs and moves into the connector. The guide slot system's ability to center and align the card and connector and to maintain the card in a central plane affects the connector's ability to provide

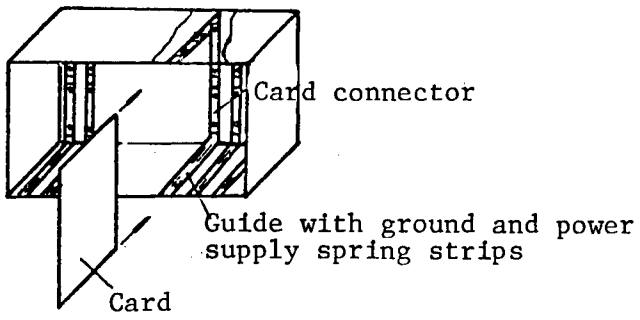


Figure 8. Board Module

reliable contact. Therefore, in designing and producing the board modules, we not only had to use special installation jigs and stringently control the precision with which the modules were assembled, but also had to use high-flexibility, low contact pressure springs as the ground and power supply springs. In this way the ratio of their clamping force on the cards to the contact pressure of the connector on the cards was relatively low. When a certain level of misalignment develops, the contact force of the connector on the card can cause it to float so that it meets the alignment requirement.

VI. Conclusions

The connectors are extremely critical components in the assembly of the computer, which have a key influence on overall machine performance. As computers develop, there is a need for continual adoption of new connection technologies and development of new connectors. How can one assure that a new product will function reliably in the computer? Our experience shows that it is very difficult for routine tests to reflect the overall picture of a product's capabilities, and we must perform thorough, detailed analysis and study of the product's material, the forming of its components, plating, assembly, and the like. In order to improve the reliability of connectors, not only must the producing plant make a stringent effort in product quality control, but the user must cooperate closely, e.g., by taking pains in rational selection and correct use of the product.

8480/9365
CSO: 4008/249

MULTILEVEL PRINTED CIRCUIT BOARDS OF 757 COMPUTER, THEIR PROCESS DESIGN

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 3, 1984 pp 51-55

[Article by Wang Keben [3769 0344 2609], Tang Jicai [0781 3444 2088], and Fang Jianbing [2455 4675 4426], Institute of Computing Design, Chinese Academy of Sciences]

[Text] I. Survey

The 757 computer is a 10 MIPS [million instructions per second] large, general-purpose computer. The requirements for its printed circuit boards are high-quality electronic performance, high density, and high reliability. The small-area multilayered boards previously used as cards, using twisted pair wires as connections between boards, and with hand-applied wiring, were no longer satisfactory, so that it was necessary to use a new board design and a new manufacturing process. Analysis and trial manufacture led to the design of a multilayer board for the 757 machine and a new production process. It has the following main characteristics: 1) A large-area 8-layer board with 72 leads is used for the cards, with 12 cards installed on a multilayer PC [printed circuit] board forming a unit; the units are connected by the wire-wrap technique. 2) CAD [computer aided design] is used for automatic wiring layout, and the boards are produced with numerically controlled photograph scanning, numerically controlled hole drilling, and numerically controlled board testing. The production line uses the adhesive film reverse plating method. The new techniques involved include ① Roller application of a positive light-sensitive emulsion; ② Roughening of the inner copper pattern layers of the multilayer board and use of presoaked epoxy dicyonomide sheet to bond the layers. ③ An adhesive film photoresist process and alkali copper chloride etching. ④ Bright electroplating with tin-lead alloy. ⑤ A multilayer board system positioning process. 4) The effect of dual in-line package [DIP] pinhole soldering and die cutting of the board outline on plated through holes was studied. 5) Quality control and testing were used during manufacture of the multilayer PC boards, resulting in an acceptance rate of up to 85 percent. This design went into production, and about 2,000 multilayer cards and about 100 boards have been produced for the 757 machine, satisfying all of the specifications laid down in advance for the computer.

II. Design of Multilayer Circuit Boards

1. Data for Design of Boards and of Manufacturing Process

(1) Satisfaction of high-speed transmission electrical requirements. In order to achieve impedance matching throughout the system and decrease power consumption, it was required that the entire machine have 90-ohm impedance matching and that the PC cards and boards have the same impedance; that the entire machine have a high-quality grounding system; that crosstalk between the signal lines in the multilayer board be kept within permissible limits; and that signal delays on the boards should be small.

(2) Solving the problem of lack of impedance matching and excessive density in past board wiring. The multilayer PC cards in the Model 013 had a characteristic impedance of 80 ohms, while the boards used twisted pair wiring with a characteristic impedance of about 110 ohms; thus the impedances of the cards and boards were not matched to each other. Use of twisted pair connections doubled the amount of wire used; after soldering, the wiring was about 10 cm thick, and even thicker at turns on U-shaped boards, which created difficulty in soldering, checking the lines and repairing them. The level of interference between lines on a board was high, and it was difficult to automate their production.

(3) Overcoming the deficiencies of the previously used hand-applied-wiring method of board production. Originally the wiring was done manually and involved a great deal of pattern application, with poor accuracy. With flow-on film application, the film clogged the holes when they were drilled, thus degrading the reliability of hole plating, and in addition the large amount of cutaway work involved in board wiring created the danger of cut-throughs, producing short circuits or exposing the glass reinforcing cloth weave so that the surface insulation resistance decreased.

(4) The 757 machine uses dual-in-line packages (DIP) whose pins must pass through pinholes for soldering; this replaces the adhesion soldering of the earlier flat IC packages. In addition, 72-lead card connectors are used instead of the earlier pin-type plug connectors, creating stringent requirements regarding the bow and twist of the boards and their thickness tolerance.

(5) The 757 machine uses small numbers of a large variety of cards; they are produced in large quantities and their production involves several difficulties, making it necessary to consider board standardization in production.

Our board design and process design focused primarily on solving the above problems.

2. Shape and Characteristics of Multilayer Cards

Card area: 175 x 278 mm^2; number of layers, 8; thickness, 2.3 ± 0.15 mm; order of layers, see Figure 1; hole diameter, 0.9 mm; ratio of board thickness to hole diameter, 3.

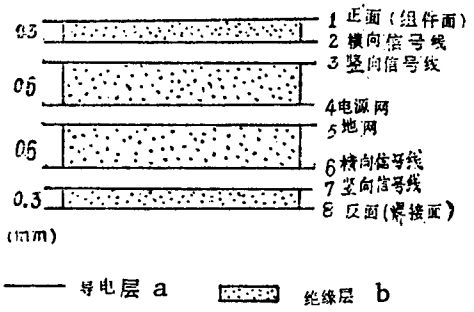


Figure 1.

Key:

1. Front (component side)
2. Transverse signal lines
3. Lengthwise signal lines
4. Power supply network
5. Ground network
6. Transverse signal lines
7. Lengthwise signal lines
8. Back (solder side)
a. Conductive layer
b. Insulation layer

Soldering pads: Outer layer, 1.7 x 1.7 mm^2, inner layers 1.5 x 1.5 mm^2.

Signal line width 0.25 mm (about 0.2 mm after etching).

Coordinate lattice spacing: crosswise 2.5 mm; vertical 3.75 mm.

Wiring density: 1 line in 2.5 mm spaces; 2 lines in 3.75 mm spaces.

Automated wiring coordinate spacing, 1.25 mm, for actual subdivision method see Figure 2(a), 2(b).

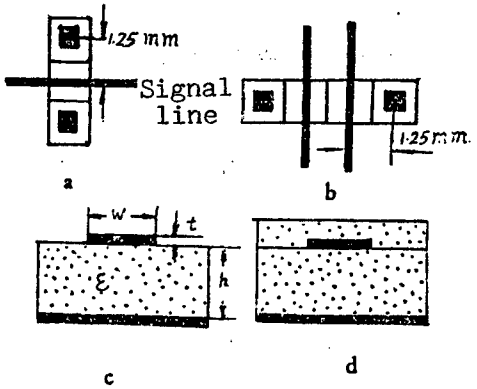


Figure 2.

Layout and dimensions of ground and power supply network layers: see Figure 3.

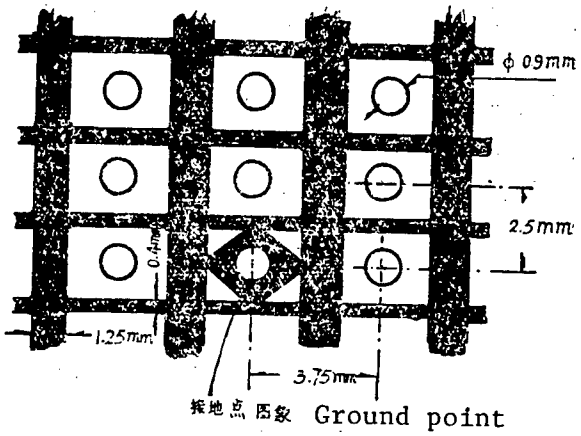


Figure 3.

Electrical characteristics: C = 80-90 PF/M (capacitance per unit length); L = 0.6-0.8 μF/M (inductance per unit length); Z_0 = 80-90 ohms (characteristic impedance); T_d = 0.075-0.08 ns/cm (no-load printed circuit delay); R_d = 0.02 ohms/cm (DC resistance of signal lines).

3. Printed Circuit Board Design Satisfies High-Speed Transmission Electrical Requirements

(1) Selection of Order of Layers, Materials, Signal Line Width, Dielectric Thickness, Ground Network Layout in Terms of Requirement for 90-Ohm Impedance Matching

The characteristic impedance Z_0 of one transmission line is

$$Z_0 = \sqrt{\frac{L}{C}}$$

L--inductance per unit length
C--capacitance per unit length

where L is the inductance per unit length and C the capacitance per unit length. When a planar microstrip transmission line is produced on a PC board, the characteristic impedance is approximately expressed by the following equation (see Figure 2(c):

$$Z_0 = \frac{87}{\sqrt{\varepsilon + 1.41}} \ln\left(\frac{5.98\,h}{0.8\,w + t}\right)$$

The equation is relatively accurate when $\frac{w}{h} \leqslant 1.25$. We chose glass-reinforced epoxy resin as the board material. Its dielectric constant is $\varepsilon \doteqdot 5.4$; and w = 0.2 mm, h = 0.6 mm, t = 0.05 mm;

$$Z_0 = \frac{87}{\sqrt{5.4 + 1.41}} \ln\left(\frac{5.98 \times 0.6}{0.8 \times 0.2 + 0.05}\right) \doteqdot 94.3\ \Omega$$

The signal lines in the interior of the printed boards are laminated in, as shown in Figure 2(d). Tests indicate that $Z_0 \doteq$ 80-90 ohms. On layers 2 and 7 the signal lines must be spaced farther from the ground by increasing the thickness of the semihardened adhesive film by 0.1 mm; accordingly the resistance of these two layers is greater than that of the signal lines in layers 3 and 6, and the widths of the signal lines in these two layers should be increased so that their characteristic impedances will be the same as those in the inner layers.

(2) Decreasing Interference Between Lines

In the sequence of layers, the four layers with signal lines in them are divided into two groups by the ground and power supply networks; in each group, one level carries the crosswise lines and the other the lengthwise lines, so as to decrease mutual interference. In the case of parallel signal lines within a PC board, there are three situations: ① the crosswise signal lines within the layer are laid out with one in each 2.5 mm coordinate space, so that the distance between signal lines is 2.5 mm and the mutual interference is low; ② the vertical lines in a layer are arranged with two in each 3.75 mm space so that the smallest distance between lines is 1.25 mm; ③ signal lines in two neighboring layers are generally staggered, producing very low interference. The few overlapping parallel lines are close together (0.1 mm), and accordingly in the automatic wiring layout process it is specified that overlapping parallel lines in adjoining layers must be less than 1 cm long.

(3) Decreasing the Length of Connecting Wires in the Computer and Decreasing Line Delay

Where component layout permits, a small coordinate grid is used, decreasing it from the original 3 mm spacing to 2.5 mm. In this way, with an equivalent number of coordinate spaces, the length of connecting lines between two points is decreased by a sixth. In addition, automatic wiring layout is used for optimal selection of the shortest path between two points, and double 72-line PC cards are used, with direct connections between the signal lines and card connection paths instead of the previous practice of running the signal lines in a circuitous path and concentrating them in a single central plug, thus increasing wiring length.

(4) Use of Multilayer PC Boards in Order To Match the Impedances of Cards and Boards, and Use of a Complete Grounding System

The multilayer PC boards measure 300 x 235 mm^2. They have the same sequence of layers and patterns as the cards, and their characteristic impedance is 90 ohms. They can hold 12 PC cards, and signal lines between cards and boards are impedance matched so that there is no wave reflection. The ground network within the cards passes down both sides and is connected to the metal spring strips in the guides and thence to the frame. In addition, the PC card has 36 ground lines connected to the internal ground of the PC board. In this way a complete grounding system is provided from card to board to

frame, and the layers carrying the signal lines are separated by the grounding network, which decreases crosstalk. The use of PC boards also gives the following advantages: their production can use automated wiring layout, numerically controlled [NC] scanning photographs, NC hole drilling, and NC card testing. The printed circuit boards are uniform, which is convenient for soldering and repair; and the board units can be assembled separately, thus overcoming the earlier deficiency of board production that only a few workers could be involved with each board and the time required was long.

4. PC Board Design Suited for Installation of the Chosen Components and Devices

The integrated circuits chosen for the 757 machine are of the DIP type and the connectors are 72-line card connectors; the choice of multilayer board characteristics must be suited to these features; ① The coordinate spacing is chosen as 2.5 mm in the x direction and 3.75 mm in the y direction, which matches the spacing of the pins within each line on a DIP package and the distance between the two lines. ② The hole diameter, ϕ, chosen is 0.9 mm, which is suited to the pin cross section of the DIP packages, and which increases the extra installation room when the size of the holes is decreased by metallization (see Figure 4).

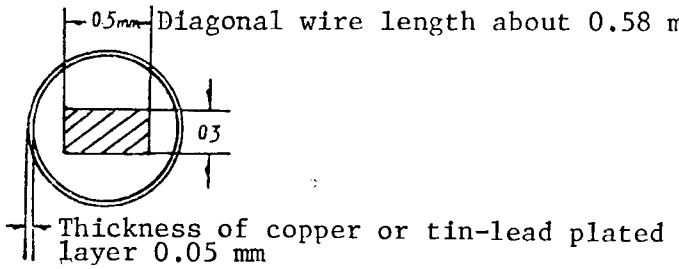


Figure 4.

The pins become thicker when tinned, but because surface tension causes a minimal increase in the length of diagonal lines, this can be ignored. ③ The board thickness was chosen as 2.3 ± 0.15 mm, primarily in terms of the requirements regarding electrical characteristics. The characteristic impedance is 90 ohms, the corresponding signal lines must be separated from the ground layer by a dielectric layer 0.6 mm thick; after the board is laminated the thickness is 2.3 mm. The card connector was chosen afterward. With a board thickness of 2.3 mm and with package pins 3.5 mm long, the pins protrude through the board about 1.2 mm after installation, which makes for ease in soldering. ④ The width of the signal lines was chosen as 0.2-0.25 mm in terms of a characteristic impedance of 90 ohms; of the factors influencing the characteristic impedance, the dielectric constant of the board material and the thickness of the copper film had already been chosen, and thus the only factors that could be adjusted were the thickness of the dielectric material and the width of the lines. Increasing the board thickness in order to increase the width of the lines would cause problems in hole drilling, and if the thickness was increased and the lines made extremely fine, the acceptance

rate would be low; this is why a width of 0.2-0.25 mm was chosen. ⑤ The choice of ground network dimensions was constained by the lattice spacing and hole diameters. Under these constraints, the largest permissible ground holes were chosen in order to increase the production acceptance rate and to decrease short circuits between plated-through holes and the ground network.

5. Board Design Suited to Production of Many Varieties in Small Quantities

In the laminated layers of the board, the solder areas on both sides, the IC package areas, and the power supply and ground networks have the same pattern on each of the eight layers for the entire machine, so that for different card models it is only necessary to change the signal line layers, and in production the photograph scanning has to be performed only once. A general purpose standardized zone was used to drill the holes for the IC pins, the plug connection holes and inspection holes. On different board models, the adaptor junction holes between layers were different, but the data for drilling them are provided by automatic wiring layout.

III. Process Design for Multilayer Board Production

1. Multilayer Board Production Process

The process flowchart is shown in Figure 5.

2. Advantages of the Process

(1) Advantages of the dry photoresist reverse plating method; ① It eliminates the possibility of paint pealing off wires and pipelines. ② The board surface quality is high, with no exposure of the reinforcing cloth weave. ③ It increases the reliability of the plated-through holes. Because the current is uniform when plating the entire board, the metal film in the walls of the holes is of uniform thickness. In addition, the board has no lacquer film, so that the problem of contamination of the holes making metallization difficult, is eliminated. ④ The use of a lead-tin alloy replacing dip metallization decreases production costs, and soldering is more reliable. This also decreases warping of the board by heat during the dipping process.

(2) Advantages of Numerical Control of Photograph Scanning, Hole Drilling, and Board Testing. ① It eliminates manual wiring and drawing of the original pattern, the automatic wiring cycle for the computer is short, the quality is high, and the percentage of completed connections is high. ② Direct numerical control of scanning of negatives gives a high positioning accuracy. The resulting negative has good black and white contrast and uniformity of pattern. ③ Numerically controlled hole drilling is done with high precision, quality is high, and labor is saved. ④ NC testing of boards makes it possible to determine correctly whether the wiring of a multilayer board has short circuits and broken circuits. In the past, with manual testing, it was very hard to detect short circuits between lines. The testing is rapid and assures that the boards are correct.

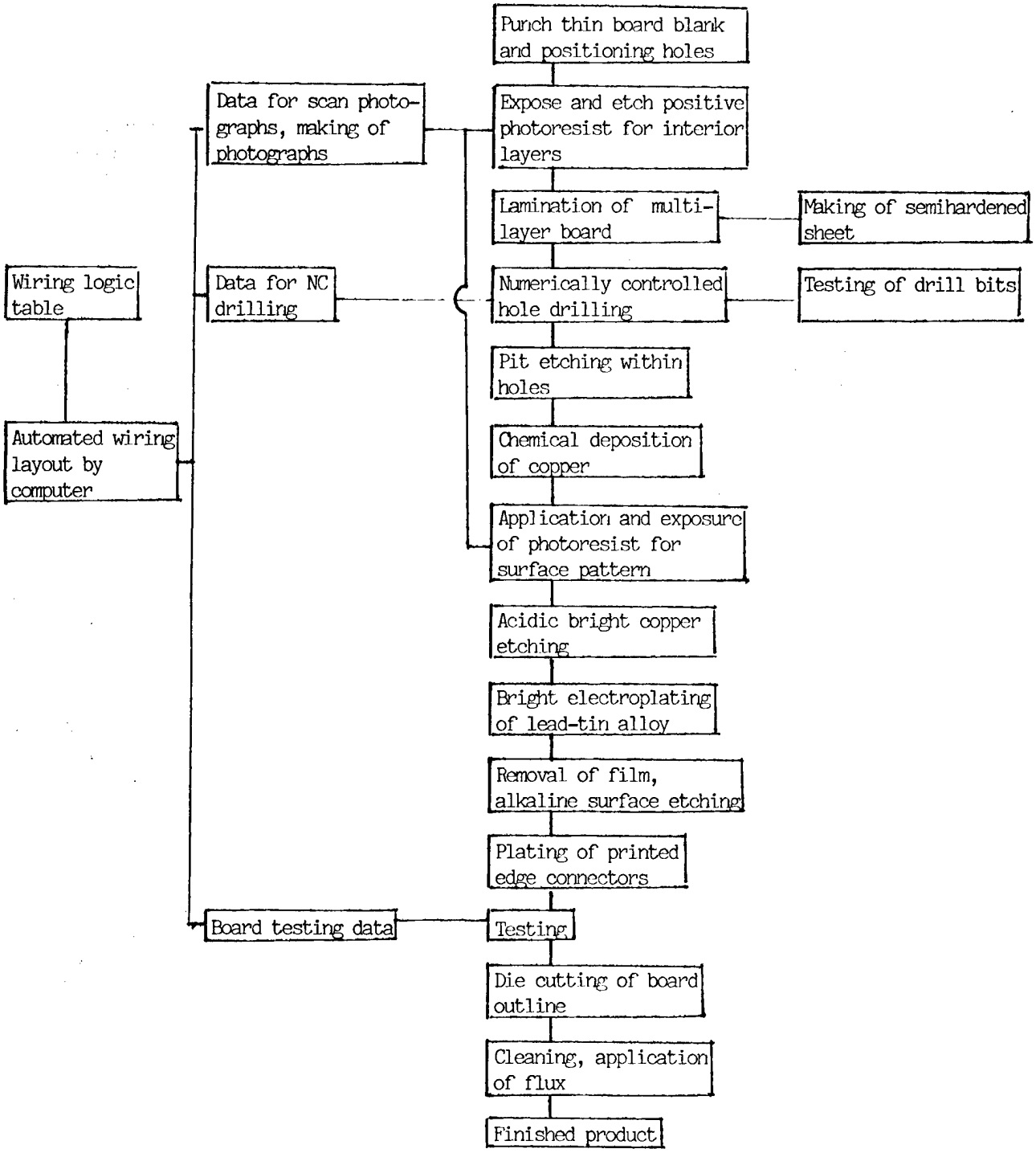

Punch thin board blank and positioning holes
Data for scan photo-graphs, making of photographs
Expose and etch positive photoresist for interior layers
Lamination of multi-layer board
Making of semihardened sheet
Wiring logic table
Data for NC drilling
Numerically controlled hole drilling
Testing of drill bits
Pit etching within holes
Automated wiring layout by computer
Chemical deposition of copper
Application and exposure of photoresist for surface pattern
Acidic bright copper etching
Bright electroplating of lead-tin alloy
Removal of film, alkaline surface etching
Plating of printed edge connectors
Board testing data
Testing
Die cutting of board outline
Cleaning, application of flux
Finished product

Figure 5.

(3) Advantages of Using System Positioning: ① The positioning holes punched in the thin film are used continually up to the die cutting stage. There is no need to change positioning holes in the middle, so that precision is high and effort is saved. ② When punching the positioning holes in the pattern, the pattern is first held immobile under pressure, then processed, which prevents it from bowing when the holes are punched, to avoid incorrect hole spacing. A pattern positioning device is used for the outer board, so that positioning accuracy is high and the pattern is not damaged. ③ Die cutting of the multilayer board assures that the dimensions of the 72-line edge connectors and the outline will be accurate.

(4) Advantages of Using Rolled-on Positive Photoresist To Produce Inner Layer Patterns: ① High line accuracy and production efficiency. ② Direct use of photographed positive pattern to scan obviates the necessity of again producing a negative, which saves on film. ③ During etching there is no excess emulsion, so that the surface does not need to be cleared, which saves machining and maintains quality. ④ Once the layer is applied the board can be stored for half a year and exposed at any time.

(5) Advantages of the Use of Semihardened Adhesive Sheet and Roughening of Interior Layer Patterns: ① The sheet has longer storage life than the anhydride form. ② It has good resistance to solder dipping. ③ It eliminates microseparations in the inner layers and increases board reliability.

3. Reliability Analysis and Control of Multilayer Printing

The reliability of multilayer boards involves board shape, material, lamination pressure, hole drilling, chemical deposition of copper, copper plating, positioning, and the like. In the past, we used concave etching; we chose a nonresin board material to eliminate microseparations during pressure lamination of the boards; we used a hard alloy drill head to drill the holes, a barium activation process for the adhesive, and acidic bright copper plating, all of which have yielded good results. The 757 machine's multilayer boards also pose the problem of the effect which soldering of the package pin holes and die forming of the external outline have on metallization. In order to assure the reliability of the boards, in addition to already existing measures, we have taken the following improvement steps, as well as performing experiments on the effect of hole drilling and soldering and blockage on quality with reference to metallization.

The reverse plating method was used instead of process wiring. This produces uniform plating thickness within the holes, and the process does not require flowing on of lacquer, which eliminates plugging of holes.

(2) Before laminating, the copper layer for the inner layers is roughened by copper chloride etching, and the contact pads for the ground network are diamond shaped, which increases adherence between layers and eliminates microseparation when holes are drilled.

(3) A glass-reinforced epoxy board is used as a pad when drilling the holes, rather than a phenolic board or epoxy-phenolic board in order to prevent

contamination during concave etching after the holes are drilled and before metallization.

(4) The resistance of the wiring holes is tested before and after the package pins are inserted in the holes and soldered, and again after the routine tests. Repeated testing during multiyear use of the machine has shown that the resistance of the holes does not change.

(5) The pin hole resistances are checked before and after die cutting of the board outline; the value is unchanged and there is no layer separation in the boards.

Large-scale production of the 757 machine's multilayer boards and multiyear operation of the entire machine have shown a high acceptance rate and high reliability. In the future, further research will be conducted on how to make better use of computer aided design and to produce high-density multilayer boards suited to large-scale integration [LSI] circuit components.

8480/9365
CSO: 4008/249

THE ARCHITECTURE, PRINCIPLES AND DESIGN OF THE 757 VECTOR COMPUTER'S ARITHMETIC UNIT

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTERS AND DEVELOPMENT] in Chinese
Vol 21, No 5, [May] 1984, pp 1-7

[Article by Shi Guohua [4258 0948 5478], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] Abstract: This paper introduces the arithmetic unit of the 757 vector machine with respect to its basic architecture, function and characteristics; skip station type and successive station type operational pipeline control under microinstruction and macroinstruction two levels of control methods; and some existing problems concerning a few major calculation methods and design.

I. Preface

The 757 vector computer is a monoprocessor with a pipeline architecture. A computer system made up of a primary (processor) and secondary (peripheral computer) computer groups, undertakes the task of the host processor, implementing primarily vector (arrays and byte strings) computation while at the same time giving appropriate consideration to scalar operational tasks.

In hardware design, the three basic technical measures, namely, time overlap, resource duplication, and resource sharing, are used to exploit fully the parallel nature of the monoprocessor system and to improve machine operating speed and system use efficiency. Improving the system use efficiency is the top priority performance target of the system design.

The central processing unit implements internally the parallel flow of the instruction level to the first order, which is divided into instruction control, memory control, operational control and the arithmetic unit, the four major functional units. The job of each unit is to complete its own fixed operation through overlapping in time.

In order to exploit fully this parallel nature and to improve the processing speed, non-arithmetic instructions are executed by instruction control while the arithmetic unit executes only operational instructions. In this way, as much as 50-60 percent of the execution time of such instructions as branch, index, GET, and loop, etc. are absorbed by the execution time of the arithmetic instructions, without affecting the output of floating point operations results and thus improving the processing speed of the host machine.

In order to reduce the possible internal collisions which may occur with arithmetic instructions addressing the internal storage, a configuration of 12 addressable multiple accumulators is used to store operands and intermediate results. The multiple cross memory technology adopted resolved conflicts in addressing internal storage and maintained a balance between busy and idle for each entity so that internal storage and the central processing unit speed could be coordinated. The design of multiple accumulators, multiple buffers and multiple parallel memories are manifestations of resource redundancy and thus clearly improve the frequency band of each functional unit.

In order to adapt the system to the demands of operational speed, the arithmetic unit uses a pipeline single arithmetic unit configuration. This type of configuration is less costly, its facility efficiency is high, and at the same time it simplifies the allocation and control of the instruction flow and data flow to the operational and control units.

The arithmetic unit deals mainly with the operation control component, and does not communicate directly with the instruction control and memory control. It carries out arithmetic operations and data editing and control. In the arithmetic unit the vector and scalar operational algorithms of the same instruction are the same with the only difference being that in execution there may be some discrepancy in the depth of the parallel overlap and operational control.

There are altogether 77 operational-type instructions in 15 classes, which, except for the prefetch instructions carried out by operational control, are executed by the arithmetic unit.

The design of the arithmetic unit was carried out based around the three targets, which are advanced nature, high speed, and reliability. In design, the characteristic functional flexibility of the ECL modules as the basic logic unit circuits is fully employed; the "wired-OR" technology and the positive/negative alternating logic technology are widely used, saving components, reducing the number of interconnected logic gates, and thus improving the speed. Such high speed algorithms as multiplace serial-parallel multiplication, iterative division, one beat polarization, normalized addition, and one beat shift are used. A detector system is set up to include auditing, double calculation, and diagnosis, thus, improving the stability of the system.

II. Design and Characteristics of the Pipeline Single Arithmetic Unit

As everyone knows, the pipeline technology divides a repetitive sequential process into a certain number of sub-processes and carries out each process simultaneously with other sub-processes at a dedicated functional station, thus implementing overlapping parallels in time. Clearly, after selecting the operational method, the important task of the designers was how to divide a repetitive sequential process, i.e., the processing instruction process, into a certain number of individual operational steps for execution. We will explain this in the example of floating point addition. Normally, addition is divided into six operational steps: finding the difference of orders, saving the larger value, symmetric value shift, summing the mantissa, standardizing the result, rounding off, and overflow processing.

Clearly, under the pipeline work mode, relying on the algorithm to divide up the operational steps demands the setting up of a dedicated function network to implement each operation. However, are these functionally different networks independently established stations, an amalgamation into one station of a certain number of dedicated networks, or is one functional network separated into two stations? This is another important question which the design of the pipeline operational component must resolve, and it is also a problem of how to divide the functional stations.

The principle of dividing the functional stations is that the execution time of the logic networks which make up the functional stations is generally approximately the same or equal. This is because only when the execution time of the functional stations is the same can the functions in the pipeline always be in the busy state so that the pipeline can complete its tasks at the highest rate of speed. Actually, the machine's clock frequency which also means that the execution time of the functional stations was decided to be equivalent to one clock cycle early in the system design.

We know that the length of the execution times of the six operational steps in addition are not the same and that there are great differences, therefore the stations could not be divided according to their natural operational steps. This meant that after the processes had been divided, they all had to be analyzed comprehensively and then synthesized.

Figure 1 illustrates the basic architectural outline of the 757 vector computer arithmetic unit. It is made up of a five-station structure. It divides the six operational steps of the addition process among four stations, i.e., difference in value codes [jiama xiangjiang [7132 4316 4161 8096]] (finding the difference in value) and saving the larger value are the first station; symmetric value shift is the second station, mantissa preload and rounding off are the third station, and standardizing the result and overflow processing are the fourth station. The result is then transmitted into the fifth station.

Although the arithmetic unit is constructed of five stations, only four stations have logical functions; the fifth station does not execute logic operation, but is only an auxiliary buffer station.

If we set aside the engineering implementation, we see that the results of the operations of standardization, shift, compressing and revising the value codes carried out by the fourth station can be sent directly to operational control L, G, and H and need not go through the fifth station. However, from the perspective of engineering implementation, this is impossible to realize for the following reasons: (1) operational control and the arithmetic unit together use six boards, the distance between them is too great, and the circuits too long so that the added burden is very heavy and there are many allocating series gates [fentuichuanmen [0433 2236 0025 7024]]; (2) the operations carried out by the fourth station are very complex and require the occupation of more logic series; (3) to send correct results, the operational results should be checked and it is impossible to implement these operations within one beat. For this reason, the E register, called the fifth station,

was created, to store temporarily the operational results of the fourth station, and buffer the tight temporal relationship between the interfaces. Adding the fifth station, generally speaking, will not influence the efficiency of operation.

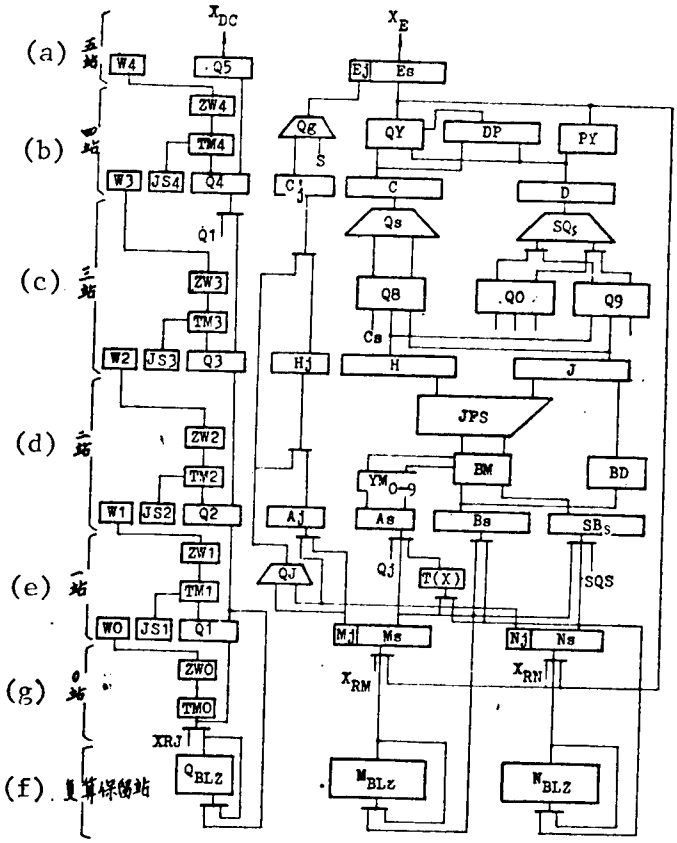


Figure 1.

Key:

a. station 5
b. station 4
c. station 3
d. station 2
e. station 1
g. station 0
f. double calculation reserve station

Each station is made up of some functionally different logic networks, such as the Qj, JFS, Qs, Qg networks. They each carry out a specially designated operational procedure of the pipeline and the execution time of each procedure takes about one beat. Registers are used as separators between stations, such as the M, N, B, H, J, C, D registers. The operational results of the previous

station are stored in the register of the next station which act as the operational number of the next station's operations. It is thus easy to see that the length of each register depends on the word length of the result from the previous station.

Looking from the angle of control, there is a 0 station preceding the first station. It is made up of two elements: one is the message sent by the Czd control station that it receives and combines to produce the conditions for setting up microinstructions and discriminates in advance the operations to be executed by the instruction; the other part carries out preselect control of commands and operations sent in the double calculation reserve station.

III. Skip Station Type and Station-by-Station Type Operational Pipeline and Their Control

Operational instructions are divided generally into:

1. Standard arithmetic operational instruction: although its functions are simple, the algorithm is very complex. It determines the basic structure and processing speed of the arithmetic unit. Implementing this instruction uses all five stations of the arithmetic unit and is called the station-by-station type operational pipeline.

2. Basic arithmetic operational function instructions for exponents, root extraction, and logarithms, such as separating an integer to exponents, which are implemented by a combination of software and hardware.

3. Pseudo-arithmetic operational instructions which perform editing operations, such as compare, shift, and logic instructions.

The latter two types of instruction algorithms are simple and can be implemented by stations 1, 4, and 5. Since they skip over stations 2 and 3 they are referred to as the skip station type operational pipeline.

In parallel tasks which overlap a great deal, how is the instruction sequence of skip station and station-by-station instructions controlled? Clearly, the control of this pipeline process is very complex and if the processing is not appropriate it can very easily lead to conflicts between devices or instruction clashes which upset the orderly process of command execution.

The arithmetic unit employs a command response control mode to control the instruction flow.

The basic idea is: the instruction about to enter the arithmetic unit is responsible for issuing a command permitting the instruction which is waiting next in sequence in the operational control execution station to enter the arithmetic unit. This command clearly stipulates which class of instruction in the execution station and when it may enter the arithmetic unit.

After the successor instruction receives a command coinciding with the conditions, it cannot directly be pushed because the operational control still

must decide whether or not the successor instruction operational operand readiness is in order. If the operand is ready, then in the next beat the successor instruction can be pushed into the arithmetic unit, otherwise, a break chain [duanlian [2451 6969]] situation occurs.

Normally, break chain refers to an interrupt in the instruction flow or to a number not yet ready which puts the arithmetic unit into the idle state. We have set up four enable to enter commands.

1. Enable primitive [yunben [0336 2609]] command

Only the vector command has the right to issue this command which is used for scheduling between components. The enable primitive command has a self-maintenance function so that when break chain occurs, it maintains its original state.

2. Enable double word high order partial entry command

To save on equipment, the double word operation uses single word equipment, thus the high order and low order enter the arithmetic unit separately. After the low order has entered the arithmetic unit, it issues a command permitting the high order to enter. The dual shift instruction permits a break chain between the high and low order operations and at this time allows the dual high command to have a self-maintaining function until after the high order enters the arithmetic unit, at which time it is released. The low order operational number waits in D, until the high order part arrives, and then one shift is carried out together.

Dual addition type instructions do not permit a break chain between high and low order, otherwise, operational chaos would occur in the arithmetic unit that cannot be checked. For this reason, after the high and low order are established to be all in order, then an "all orders arrived" ["shudaogi" [2422 0451 7871]] condition is issued.

3. Enable to add command

Only addition and divide commands can issue enable to add commands, and only addition, multiplication and divide commands which are ready in the execution station can accept and execute enable to add commands.

4. Enable new command

Its rank is high and its functions include: (1) It is the arithmetic unit start command, and in the initialization stage it is used to schedule the instruction flow of the first instruction to enter the arithmetic unit; (2) during break chain, it is used to restore the instruction flow; (3) after an enable new command is issued, any instruction in the execution station must enter the arithmetic unit in the next beat as long as its numbers are in order.

This type of command mode control of the overlapping of the instruction flow is effective and it works. They systematically schedule the instruction flow

in the arithmetic unit sometimes in serial and sometimes in parallel and thus exploit the feature of a lesser amount of equipment in a single arithmetic unit but not a lower rate of pipeline efficiency.

IV. Algorithm

1. Floating-point addition algorithm

This operation is carried out in five beats and is characterized by implementation of polarization and standardized operations within one beat. For example, (A) + (B) → C, in which A (a_0, a_1,...a_n), B(b_0,b_1,b...b_n), C(c_0,c_1,...c_n) are all vectors. Actually, the arithmetic unit carries out a series of additions, $a_0 + b_0 \rightarrow c_0$, $a_1 + b_1 \rightarrow c_1$, ... $a_n + b_n \rightarrow c_n$. The unifunctional pipeline [dantiao [0830 2742]] addition flow is as below:

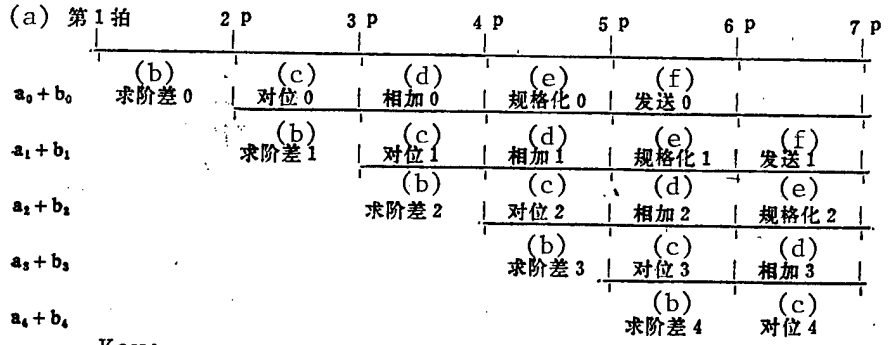
First beat to subtract exponents and save the large value: the instruction receives Q_1, a_0 and b_0 receive M and N respectively, and through Qj to find $\Delta j = Mj - Nj$;

The second beat polarization: if Δj is positive, M is the large number, Ms sends to SBs, and Ns sends to Bs; if Δj is negative, N is the large number, Ns sends to SBs, and Ms sends to Bs; Bs shifts right $|\Delta j|$ places through JFS.

The third beat mantissa addition: SBs sends to J, the number shifted through JFS sends to H, H, and J through Qs.

The fourth beat standardization: the sum is entered in C, discriminating standardized places is used to turn on the QY network's shift gate and through Qg revise the resultant exponent.

The fifth beat transmits: the result which has been standardized is stored in E so it can be transmitted. Figure 2 illustrates the addition pipeline's overlapping processing time.



Key:

a. first beat
b. find difference
c. polarize [duiwei [1417 0143]]
d. add
e. standardize
f. send

Figure 2. Addition pipeline overlap

2. Multiplication algorithm

To improve operational speed, a multidigit rapid multiplication algorithm, which still uses the idea of addition-shift loop for multiplication, was used. The multiplication execution process can be viewed as an addition process of a series of addends.

As concerns the application of the SSI integrated circuit to make up a large scale rapid system, the multidigit string parallel multiplication is still rather advanced.

The multiplication unit design developed mainly along three lines:

(1) Reducing the number of addends: this computer uses multiplication by groups based on three, it consolidates three multiplicands of a certain multiple which were to have been processed and reduces them to one addend. For example, 56 x 56 is divided into two processes, the first carries out 56 x 29 iteratively and the second carries out 56 x 27 iteratively. Then add the difference of the two products of places to obtain the complete answer.

(2) Formation of the accelerated addends: the right 29 places (or left 27 places) of the multiplier send in the encoder to be encoded, using gate control the multiplicand multiple forms the circuit BM so that BM simultaneously outputs 10 group (or 9 group) addends. The multiple gating network structure is simple and designing a rapid encoding circuit is the key to accelerating the formation of the addends.

(3) Accelerating summation of addends: we designed a four level structure adding tree permitting the addition of 9 addends made up of memory carry full adders. Its word length is 84 bits and each bit is made up of 7 full adders. Using positive negative alternating logic design technology, partial sums and carries have at most seven level delays, addition sums summation is still relatively fast. Figure 3 illustrates the four level structure addition tree.

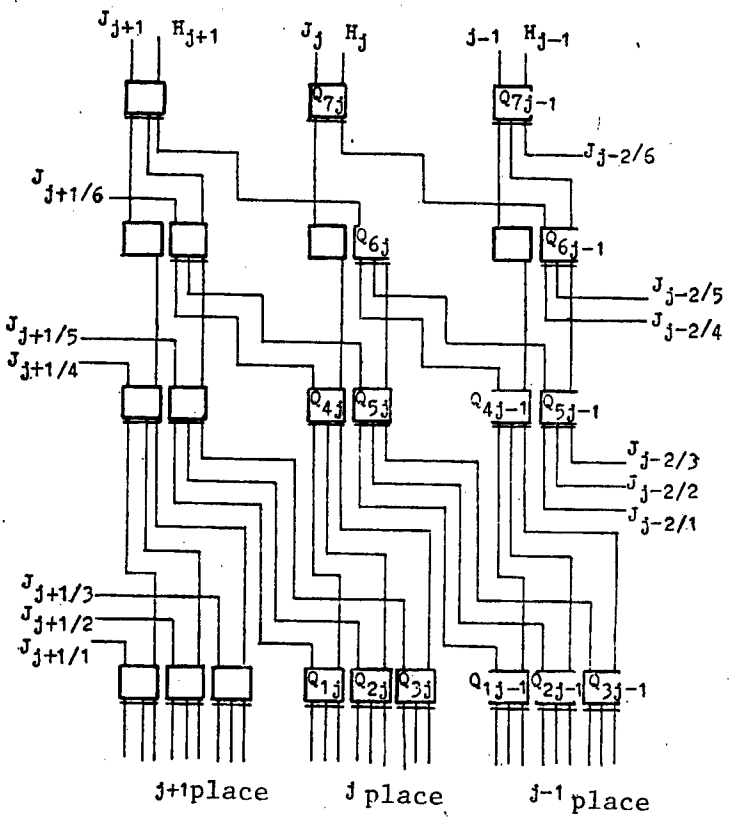


Figure 3 Addition tree

The multiplication formula derivation result is:

$$Cs = As * Bs$$
$$= Csf + Css$$
$$= Csf + Css_2 + 2^{-27}Css_1$$

in which $Csf = Asf \forall Bsf$ is the product symbol; $Css = Css_2 + 2^{-27}Css_1$ is the complete multiple; Css_1 is the first product; Css_2 is the second product.

On the basis of the above formula, the decoding formula is:

$$-A_1 + \frac{1}{2}A_2 + \frac{1}{2^2}A_3 + \frac{1}{2^2}A_4$$

It is easy to see that it will translate 16 states, and eight operations, i.e., $\pm$ Bss, $\pm \frac{1}{2}$Bss, $\pm \frac{1}{4}$Bss, and $\pm \frac{3}{4}$Bss. As soon as multiplication begins, the 3/4 times multiplicand should be formed in preparation for decoding the gating. The multiplication process is generally as follows:

1. Finding Css_1:

t = 0, multiplication instruction enters $Q^{(1)}$, multiplier and multiplicand enter M and N respectively;

t = 1, Ns sends into Bs, SQs implements ¼Ns + ½Ns addition and sends into SBs. MS right 29 digit sends into As, 10 group encoder decodes separately, each group interpretation gates one group multiplicand multiple forms a circuit, takes the addition sum obtained in order of difference of 3 places, obtaining two numbers, the partial sum of Css_1 and a carry:

t = 2, the above two numbers are stored in H and J;

t = 3, find Css_1 = H + J, save in C and D.

2. Finding Css_2:

Css_2 begins execution from t = 2 and is carried out overlapping with Css_1. At this time Ms left 27 digits send into As and the second decoding, it is the same operation as in t = 1, is carried out. t = 3 saves the partial sum obtained and advanced to store in H and J.

3. Finding Css:

When t = 3, Css_1 is in C and D, and Css_2 is in H and J; t = 4, Css is found through Qs and SOs, Css is a double word product and the word length is truncated in line with the demands of the instruction; t = 5, Css is entered into Cs and standardization is carried out; t = 6, the result is sent to E and the next pulse is emitted.

Figure 4 shows the overlapping process times of the multiplication instruction. From the diagram it can be seen that general multiplication takes only 5-6 beats. When not interrelated, it takes only 2 beats to produce a result. When the pipeline is stable and the equipment is not idle, the utilization rate is at its highest.

V. Discussion of Some Questions Remaining in the Arithmetic Unit Design

Software debugging and test problems over the past two years have demonstrated that the design of the 757 vector computer's arithmetic unit is successful and its primary functions reach or surpass the demands of the anticipated technical indicators. Under high efficiency conditions, the fixed speed of the arithmetic unit can achieve 8.2 million floating point additions per second or 4.1 million floating point multiplications, or 10.25 [1.025?] million floating point divisions. In the design process, to the extent possible, many advanced design techniques and high speed algorithms were used and much beneficial experience was gained, but there are still some inadequacies. The aim of this study is to learn and to provide reference points for improving and developing similar vector computers.

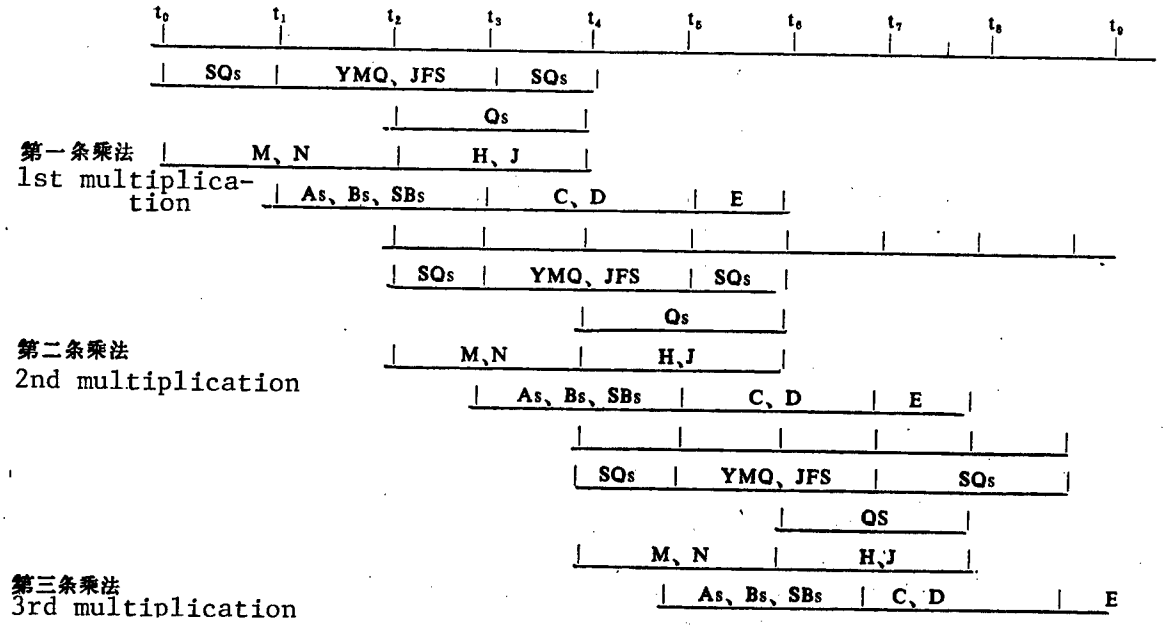


Figure 4 Diagram of time overlap in multiplication pipeline

1. Questions of introducing partial operations in the parallel structure of the pipeline

Previous arithmetic unit designs used the concept of shifting to partial operations customary for earlier traditional computers for normalization and shift operations exceeding 8 bits. Local operations take place mainly in the fourth station. When executing a process that shifts to local state, the pipeline encounters a serious obstacle. At this time, each pipeline function station behind the "local point" rapidly becomes empty and is in the idle state; each pipeline function station in front of the "local point" becomes filled consecutively, but there is no way for the flow to get through and most are in the waiting state. At this time, besides the "local points" being in a very busy state, the rest of the pipeline function stations are "asleep" and only can be awakened after the partial operations are concluded and control is returned to the center. It is obvious that local operations not only clearly lower the operational efficiency of the pipeline, but also make control very complex. Since local conditions both lock up the first, second, and third stations ahead of the "local point" and lock up station 0, over half of the arithmetic unit and the operational parts of the execution stations are nearly locked up, its burden is very heavy. This not only increases the drive levels added in the control link by at least two to three, but also radiates several hundred leads in all directions from the "local point" which is very hard to accomplish in high speed projects.

Through analysis and research we learned that we cannot apply the local control concept in the pipeline structure mechanically. The problem that local

control creates is not a general "bottleneck", but a basic problem of creating an "appetite" that's too small and cannot process data at high speed. Thus we have revised the design to make normalization and shift control one beat operations so that the flow would be unobstructed and under stable conditions achieve a maximum operation rate of one output task per beat.

2. Concerning station-jump operational pipeline

Arithmetic unit adopted station-jump and sequential-station type operational pipeline modes for processing of instructions. Viewed narrowly, this type of set up can make the non-four fundamental operations of arithmetic instruction flow jump the second and third stations and go directly to the fourth station as if to accelerate the flow process and improve flow efficiency.

In fact, this is a false impression. This is because operational efficiency is not decided by the number of pipeline stations, but mainly by whether or not enough tasks can be provided to fill the pipeline, reduce idle and waiting time, and strive to be able to output a result during each cycle.

Using the instruction sequence formed by instructions of these two kinds of flow makes scheduling and control very complex. It creates problems in the design of operational control components, adds many devices and also increases the number of levels in the logic chain. If only the sequential station flow mode is adopted, enable commands could be greatly simplified and control be made much simpler, yet flow efficiency would not decline.

3. Concerning micro-instruction control mode

The arithmetic unit adopts micro-instruction and macro-instruction control. Macro-instruction control is normal combination logic control and is control of combined implementation of a beat decoding and the conditions it generates. In the arithmetic unit, this analytical and synthetic control process generally requires only 6-8 logic levels.

Micro-instruction control predicts the conditions created by an instructional operational step one beat in advance, discriminates the combination of these conditions in advance and stores them for use by control in the next beat. Its superiority is that it moderates the shortness of time of the long chain control group used by that beat and shifts the pressure to the logic link in the previous level letting the previous beat absorb it and lighten the pressure on the time used by that beat. The logic designers frequently adopt this type of design technique to compress or reduce the number of long chain logic levels. Under appropriate circumstances, using a smaller number still can be effective.

In short chain control design, the group conditions formed by a beat can be controlled in a timely fashion thus it is not necessary to adopt the prediction and prejudgement techniques in the long chain. Advancing prejudgement is not only very complex, but also very expensive. That is to say, from the angle of control, microinstruction control mode does not have any superior characteristics and advantages. As far as trying it out in other aspects, the advantages and disadvantages will have to be weighed.

4. Where are the arithmetic unit's "bottlenecks"?

Analysis and practice shows that the triple number adder SQs is an arithmetic unit "bottleneck". SQs is mainly used for multiplication and division operations seeking triple numbers. When the multiplication/division pipeline is operating, it is constantly in the busiest state but some function networks are nearly in a semi-idle waiting state, equipment frequency band loses its balance which affects operating efficiency. If in improving the model we are willing to make some outlays, add other equipment and readjust the operating process the efficiency clearly might be higher.

The process of developing the 757 vector computer arithmetic unit was tortuous and uneven. Its successful development was the result of collective effort and is the crystallization of the wisdom of the masses. In the early stages of the arithmetic unit logic design (July, 1975-September, 1977) Comrades Wang Peixian [3769 0012 7359], Tian Gongxing [3944 0361 5281], Fu Chaoyuan [0102 2600 0337], Li Xiuying [2621 4423 5391], Chen Dingxing [7115 1353 5281], Zhang Fujiang [1728 4395 3068], Xu Jun [1776 3182], Xu Kunming [1776 2492 2494] and Man Yunxia [3341 6663 7209] participated in part of the design. In the later logic design, project design, linking and debugging test calculations and reliability testing stages, Comrades Wang Zanming [3769 6363 2494], Li Minfu [2621 3046 3940], Li Xiuying, Li Ceming [2621 4595 3046], Chen Dingxing, Yang Yucheng [2799 7183 2052], Chen Hongan [7115 7703 1344], Hou Qi [0186 0796], Xu Kunming, Zeng Fulin [2582 0102 2651], and Liao Qingyu [1675 7237 5940] also participated in the work.

8226/12712
CSO: 4008/364

ON SOME PROBLEMS IN THE DESIGN AND DEBUGGING OF THE INSTRUCTION CONTROL PULSE SYSTEM OF THE 757 COMPUTER

Beijing JISUANJI YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese No 8, 1984 pp 1-7

[Article by Song Chengjiu [1345 2052 0036] of the Beijing 9th Research Institute, Wang Zongpei [3769 1350 3805] and Teng Chunming [3326 2504 2494] of the Institute of Computing Technology, Chinese Academy of Sciences]

[Text] I. PROBLEMS

The pulse cycle, logic chain length, and transmission overmeasure in the pulse synchronous system of a large scale computer should satisfy the following relations:

$$T = n \cdot t_{pd} + l \cdot t_{dl} + \Delta$$

in which: n is the number of logic gates through which the signal passes in a given cycle, or called the number of levels for short.

t_{pd} is the time required for the signal to pass through one gate.

l is the length of the transmission line through which the signal passes in a logic chain, or called line length for short.

t_{dl} is the time necessary for the signal to pass through one unit of line length.

$n.t_{pd}+l.t_{dl}$ is the length of the logic chain.

Δ is the transmission overmeasure, or called overmeasure for short.

This paper was received in Feb 1984.

A certain overmeasure is provided to overcome the lack of uniformity created by a large number of components in the system, many routes, scattering and drift of component parameters, unevenness of routing lengths, and changes in pulse frequency and at the same time it is also to avoid unexpected situations which occur in the design and debugging of large scale computer pulse systems.

Table 1

Item	109 B	111	013	757 instr. ctrl*	MU-5	ILLIAC-III
Cycle	330 ns	330 ns	167 ns	100 ns	50 ns	67 ns
Number of design levels	6	10	25	17.5		
Number of actual levels	6	10	27	17.5	8	
Designed level delay (with negative load)	40 ns	25 ns	5 ns	4 ns		
Actual level delay (with negative load)	30 ns	20 ns	3.5 ns	3.5 ns		
Actual level delay (with negative load and route)					4 ns	1.5-5.5 ns av. 2.5 ns
Routing length	5 m single line	5 m single line	5 m twisted pair line	4 m twisted pair line		
Designed delay time	255 ns	265 ns	165 ns	90 ns		
Actual delay time	195 ns	215 ns	134.5 ns	<100 ns		
Designed overmeasure	23%	20%	17.5%	18%		
Actual overmeasure	41%	35%	20%	<10%	20-30%	25%

*Refers to 757 computer instruction control when operating at T=100 ns.

There are not yet any standardized computational formulas for selecting number of levels, line length, and overmeasure so they are based mainly on the experience of the designers.

In the process of designing the 757 computer, it was stipulated that within one pulse (T=100ns) the number of logic transmission levels permitted would be 17.5, routing length would be 2 m, the delay after loading each gate would be 4 ns, and the delay for each meter of routing would be 6 ns. Thus,

$$\Delta = T - t_{pd} \cdot n - l \cdot t_{dl} = 100 - 4 \times 17.5 - 2 \times 6$$
$$= 18 \text{ ns}$$

i.e., Δ takes up 18 percent of the cycle.

There have been various ways of considering whether or not determining number of levels, line length, and overmeasure in this way is a rational exercise. For example, if we compare the overmeasure determined for the 757 computer

with similar class computers from China and abroad (see Table 1), it is clear that the overmeasure of the 757 computer is small, and thus, can the design index determined be implemented in this way?

II. BASIC DEMANDS OF PULSE SYNCHRONOUS SYSTEMS ON D-TYPE TRIGGER STRUCTURE

Information transfer in a synchronous logic circuit composed of D-type triggers can ultimately be simplified to transfer of information between register--combinational logic circuits--register (see Figure 1), i.e., the signal is emitted by the source register, and is relayed to the target register through the combinational circuit.

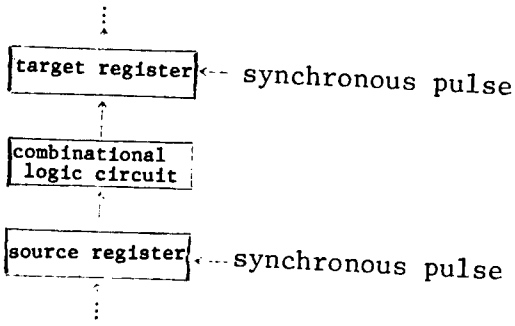


Figure 1

The register is made up of triggers, and for the D-type trigger to operate normally, the fixed intrinsic component properties demand that we do the following:

1. Before the synchronous pulse reaches the action side, the signal to be sent to the trigger should reach the input end. This is what is referred to as the set-up time $t_{set\ up}$ (see Figure 2(b)). In the figure a "1" signal must be sent to the trigger thus, before $t_{set\ up}$, the signal should be at level "1" (high level in this figure).

2. After the action side of the synchronous pulse, the input signal still must be maintained for a time t_{hold}, otherwise the trigger stage set up may not be correct. This is the hold time. Only after passing the hold time t_{hold}, can the signal "1" in Figure 2 (b) be cancelled and thus after the synchronous pulse action side, the trigger can be held in the "1" state.

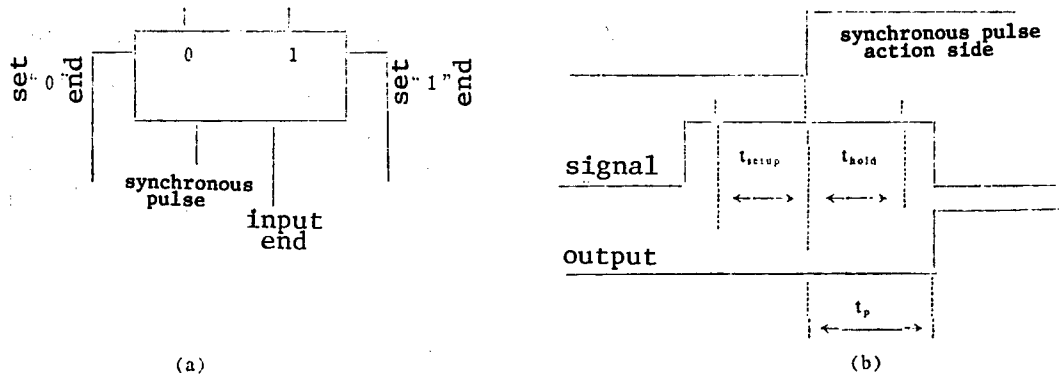


Figure 2

3. While satisfying conditions 1 and 2, after the synchronous pulse action side has, after a certain time t_p, the trigger's output can receive the necessary signal. t_p is the trigger transmission delay time.

Thus, to ensure normal transmission of the signal the following is necessary:

$$T - t_p - t_{pd} \cdot n - l \cdot t_{dl} - \Delta \geqslant t_{set\ up} \cdots\cdots \quad (1)$$

$$t_p + t_{pd} \cdot n + t_{dl} \cdot l \geqslant t_{hold} \cdots\cdots \quad (2)$$

Formula (1) means that after going through trigger delay t_p, gate delay $t_{pd} \cdot n$, routing delay $t_{dl} \cdot l$, and taking into consideration a certain overmeasure Δ, the signal can still reach the trigger input before $t_{set\ up}$, and for this reason the logic series and routing length should be provided and a certain overmeasure should also be provided. If the demands of formula (1) are not satisfied in the design and implementation process, signal relay errors will occur. Formula (2) means that the signal should not be terminated before t_{hold} has ended since $t_p > t_{hold}$ in D-type triggers, thus even when a trigger sends the signal directly to another trigger without going through a logic link, i.e., when $t_{pd} \cdot n + t_{dl} \cdot l \approx 0$, the demands of formula (2) are also satisfied.

However, when the synchronous pulse is not completely registered, such as when there is a backshift in the synchronous pulse of the receiving end, the signal may be terminated before t_{hold} is concluded, destroying the demands of formula (2), creating an information relay error and this is a so-called "catch-up" problem.

III. PROBLEMS ENCOUNTERED IN DESIGN

Instruction control is a constituent part of the 757 vector computer. Instruction control carries out the functions of call instructions, access instructions, analysis and machine-instructions, execution part instructions, entry or exit interrupt, and entry double calculations and monitoring.

Instruction control uses 8628 block modules, 233 block cards, and 8140 circuits.

The entire instruction control is packed on two baseboards 1640 x 524 mm.

In the instruction control design process there were problems concerning the logic chain, primarily two in number: one was the number of logic levels actually useable, and the other was the routing length. These problems will be presented below:

1. Number of logic levels actually useable

(1) Pulse formation and distribution [fentui [0433 2236]]: Implementing the instruction control's beat control operation method demands the formation of control pulses M_0, M_1, M_2, M_3, M_{j1}, M_{j2}, M, M*. The machine's operational state determines which of these pulses is to be issued. The control instruction working pulse serves as the source for these pulses.

These pulses are the synchronous pulses of the trigger components. Since the number of synchronous points is large and there is a limit to the load ability of the pulse distributor (there is one load board for each pulse output line), an entirely synchronous working pulse can be formed only after going through two or even three levels of distribution. Thus, from the instruction control source pulse M_3 to the formation of the synchronous point control pulses takes time for spot use of 4-5 levels.

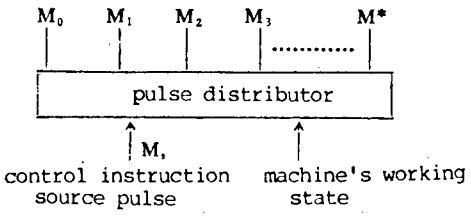


Figure 3

(2) Lockout of pulse: In the instruction control working process, there are two reasons for demanding a stop work pulse.

The reason for first type of lockout pulse is production of a correlation in the operational process of instruction control which requires a lockout pulse to wait until after the correlation has been dissolved before continuing work.

The reason for the second type of lockout pulse is detecting trouble in the operational process of instruction control. At this time, the issuance of the next working pulse should be locked out on the beat in which the trouble occurs so as to retain the current state and roll it in for double calculation or so as to retain the current state for manual or diagnostic program investigation of the error.

Because there are many (86) relational lockout pulse and trouble lockout pulse points and the number of component fan-ins is limited, it also takes 2-3 levels for the bus signal to form the lockout pulse.

From (1) and (2) we can see that in the 757 vector computer the number of logic levels that can actually be used is around 10 (9.5-11.5).

2. Measures adopted in design

The following measures were adopted to design the combinational logic circuits within the range of 10 levels:

(1) Number of compression levels: find the combinational circuit which is long in number of levels and compress it. For example, the circuits which formed the sums of the address adder was reduced from the original 5.5 levels to 5 and parity formation from 11.5 levels to 7.5. Of course doing this increases the volume of equipment correspondingly.

(2) Changing working mode: changing the link and working modes which were insufficiently rational to begin with to reduce the number of levels.

For example, the original design included semiconductor memory which had a read-out register. This required one beat after the read-out semiconductor memory address was given (i.e., the time from when the synchronous pulse A was issued until synchronous pulse B was issued) for the read-out contents to reach J_z.

The original working mode used the concept of magnetic core storage and did not take into consideration the characteristics of semiconductor storage, i.e., that if one wants to give a specific address then one can obtain a stable output, and added an unnecessary register J_{dz} between the J_d and J_z registers.

After improvement, the J_{dz} readout register was eliminated and this both reduced the number of levels and saved on equipment.

(3) Rational determination of check time: The parity check circuit is a combinational circuit. Its output can change at anytime but only during check time does the parity check output have any meaning and thus for some multiple beat working combinational circuits, the last beat of a multiple beat should be checked to avoid incorrect polarity. For example when executing a transfer instruction, the address adder first should form the transfer address, then the discrimination circuit should determine whether or not this address is outside the limits.

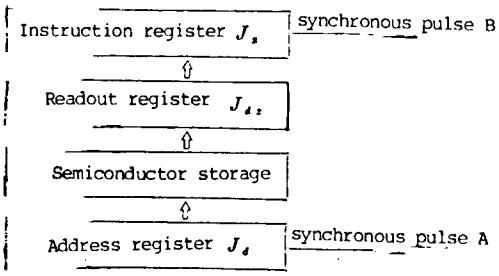


Figure 4

These two operations take two beats of logic. For this reason, check time should be placed at the end of the second beat because this is rational. Otherwise, checking each pulse may cause misrecognition of an operation between two beats and this is irrational.

3. Routing Length

Routing length is directly related to the number of components used, the packing density, and the structure of the base. If few components are used, packing is dense, and baseboard structure is tight, then routing length will be short, otherwise it will be longer.

Instruction control uses 8628 components (not including spares) and 233 cards. Even when the baseboard configuration is dense (4724 circuits/m^2) it is very difficult for each component link to be less than 2 m. The following measures have been adopted to resolve this problem:

(1) Compact assembly has been adopted so as not to increase the number of baseboards: In the late stages of design, when completing the instruction control definition of functions, we did our utmost to control and reduce the number of components and cards used, with the aim of compressing the entire instruction control into two boards to reduce routing length.

The result of compaction was that the two boards had a total of 240 card positions of which 233 were used, including one pulse source card which had to occupy four card positions because of manual adjustment frequency. Thus, there were two empty card positions on each baseboard.

(2) Division of subcomponents: Depending on function, instruction control was divided into eight subcomponents: call instruction, fetch instruction and analysis instruction, address operation, change address and change buffer control, interrupt and uniform numbering register, recovery and monitoring, pulse allocation and pulse lockout, and control and display. These subcomponents all have a certain degree of independence, there are many links between them, few links outside them, and when installed on the baseboard, these subcomponents occupy an independent continuous area to reduce the routing length between subcomponents.

(3) Determining the priority arrangement sequence of the baseboard: When dividing up the baseboard, a plan is adopted based on the relations between subcomponents and the link relations of instruction control and other control components. This plan makes access instructions, analysis instructions and pulse distributions to be the core; call instructions, address operations, change address and buffer control to be the middle; and interrupt and uniform numbering registers, recovery and monitoring to be the outer distribution scheme. Thus, the layout takes into consideration the number of natural links between subcomponents, reduces the routing length of the core and the middle sections, and also uses the characteristics of short logic levels of peripheral subcomponents which can relax routing demands.

(4) Compressing the number of levels in instruction control interface leads: The short-time problem introduced by shortening the length of leads between the instruction control and other control components demanded that at the time of design the instruction control output signal be generally permitted to pass the distribution gate only after entering the trigger, immediate output is not permitted to pass through the combinational logic circuit and then be output. Individual output signals which had to pass through logic combination and which were also long in number of levels should use trigger separation. As concerns signals input to instruction control, they generally did not go through logic combination after reaching instruction control and were only permitted to be immediately used or entered into the trigger after going through distribution to be saved.

The number of logic levels and the routing lengths of some typical instruction control logic links after adopting the above measures are given in Table 2:

Table 2

Selected logic chain number	1	2	3	4	5
Original number of logic levels	16	16	14.25	18	17
Original number of cards	7	6	5	6	8
Length of link between cards (m)	2.27	3.78	3.49	2	2.9
Length of link within cards (m)	1.44	1.95	2.00	2.28	2.17
Total length of original circuits (m)	3.71	5.73	5.49	4.28	5.07
Time of original logic chains	95.1ns	111.2ns	106.6ns	109.4ns	111 ns
Number of logic chain levels	15	15	13.25	17	16
Number of cards	6	5	4	5	7
Length of links between cards (m)	1.79	3.12	3.69	1.49	2.15
Length of links within cards (m)	1.3	1.78	1.82	2.1	1.95
Total circuit length	3.09	4.9	5.51	3.59	4.10
Logic chain time	86ns	100.7ns*	98.2ns	99.9ns	99.27ns
Measured logic chain time	87ns	73ns	83ns	92ns	92ns

* Number is an estimated value. There is a statistical error creating a large discrepancy with the test value.

IV. PROBLEMS ENCOUNTERED IN DEBUGGING

The methods and measures we adopted in the stage of debugging the frequency range were:

1. Test procedure

To inspect the frequency range strictly we wrote a 36 segment control instruction inspection program. Each segment had 64 statements and each segment program operational cycle was 150-600 pulses so that the oscilloscope viewed the wave form detail of each beat.

2. Frequency range standards

Debugging the correctness of instruction control logic links is done at low frequencies (under 1 MHz). In this way, trouble caused by electronic technology and components with poor characteristics can be avoided so that the debugging is limited to the range of correct logic. Also, the goal of frequency debugging is, on the one hand, to eliminate operational errors caused by technology and components with poor characteristics, and, on the other hand, to measure the highest operational frequencies. In a large-scale pulse system, because of the extraordinary complexity of the signal relay paths and combinational situation, the correctness of operation cannot pass as standard at a set frequency by several segments of inspection program. For example, when the logic chain through which a group of signals is going exceeds the clock cycle, it is possible that trouble with this group of signals cannot be discovered at the check points. When the frequency is low it is possible to discover the trouble with this group of signals at the check point. This is commonly referred to as the phenomenon of "able to work at normal frequencies but not able to work at variable frequencies."

For this reason, determination at the time of frequency range debugging, requires a comprehensive 36 segment inspection program under continuously changing conditions from low frequencies (single pulse) to high frequencies for normal operation. The working range determined in this fashion is called the working frequency range. Practice proves that after debugging such a range and putting it into operation there are no reoccurrences of hidden frequency problems.

3. Trouble shield and trouble shooting

There are two ways of determining whether or not a test program is operating normally when the frequency is changing. One is to set a program comparison point (a soft check point) in the inspection program. For example, if the results of the comparison point and the anticipated results are not the same, then the machine turns to generalized shut-down state showing that the signal relay path at this frequency is not clear or there is some parity check error in the relay process. The other one is to use the parity check point built into the computer. If a parity error occurs in the debugging process of working frequency range, then the lockout pulse holds the current state. This means that the logic chain is too long for this frequency and it can no longer guarantee normal relay of the signal between registers.

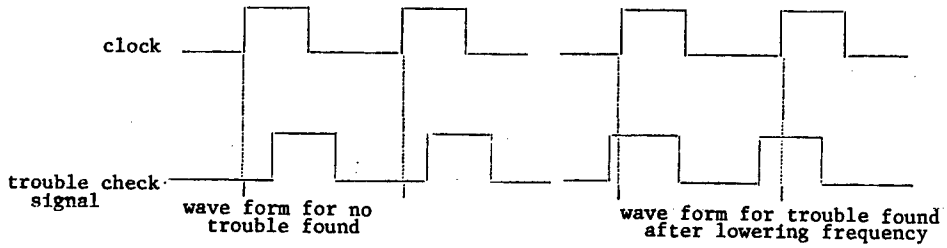


Figure 5

In the process of actual frequency debugging, when the frequency is made higher if a parity error lockout pulse occurs, this also means that the parity check point set-up has basically overlaid the instruction control.

Under such conditions it is possible to pursue the source using the signal of the trouble lockout pulse. When the frequency is steadily changed from low to high we look for a lockout pulse signal to appear. If a lockout pulse signal appears, then it can be pursued level by level until the source of the trouble is found, that is until the long logic chain is found at the given frequency. This is the troubleshooting method. This method goes step by step using an oscilloscope and is rather troublesome.

The trouble shield is a less troublesome method. The basis of this method is that each problem can be shielded. If a problem appears at high frequency and the computer is stopped, it requires only the use of a single probe to test the shield point by point and after the shield points have been selectively paired, the computer is restarted and the inspection program can again operate normally. Thus the shield points facilitate our finding the source of the trouble.

Troubleshooting and shielding can be used in combination. In this way the random nature of troubleshooting on a large-scale pulse system can be reduced.

Results of Debugging

After eliminating frequency trouble created by trouble in the primary components, the maximum operating frequency of the inspection program is 8.2 MHz (cycle is 122 ns), and this is the overall result of logic design, engineering design, and structure design.

So that it can operate at working frequencies below 10 MHz, in addition to the above measures, we also used the concept of pulse system quasi-registration as a supplementary measure.

In the second section of this article we have already assumed that the time at which synchronous pulses get to the register is equal, that is, they are aligned, thus, for signals transmitted between any two groups of registers, the

cycle time is an equal value T, but the number of combinational logic circuit levels between any two groups of registers is not necessarily equal, thus when raising the working frequency, for certain signals, the number of logic levels is clearly tight and can even cause trouble. For some other signals, although the frequency is raised, the number of logic levels still has an overmeasure and it is just because of this difference that we can hold back the synchronous pulse of a long chain's target register (if the next level's logic chain is not tight) and lengthen the cycle of a long logic chain, at the same time reducing the cycle of a short logic chain so that trouble is eliminated and the overall working frequency is increased, as illustrated in Figure 6.

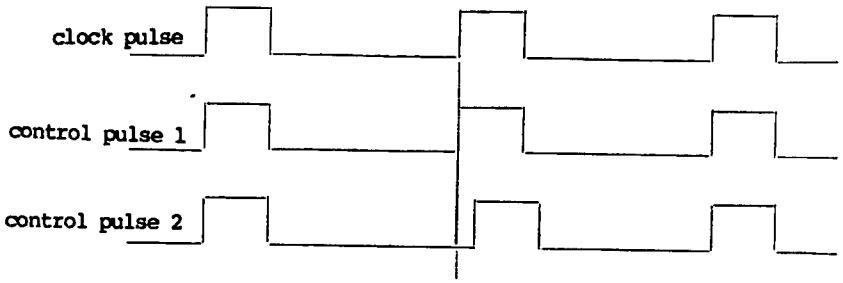


Figure 6

After the quasi-registration measures were adopted, a total of 13 logic chains (7 pulses + 5 potentials) were adjusted so that the instruction control working frequency reached 11.2 MHz (T=89ns).

It should be noted that this quasi-registration method can only serve as a supplementary method and can only be used in the debugging process of similar computers.

Using the above methods was only to satisfy the frequency norms and demonstrating the tightness of the logic chains in the original design.

V. CONCLUSION

The number of logic levels actually used is related to the complexity of the synchronous pulse system, the pulse distribution mode and the lockout mode. When the registers are limited to a single level structure the number of logic levels actually useable determines the quantity of components. Reducing the number of useable levels increases the components used. Thus when determining the number of logic levels it is necessary to balance several factors otherwise contradictions will appear and lead to a lowering of the working frequency.

2. Routing length is related to packing density and the number of components used. In a large scale computer with the packing configuration and density similar to that of the 757 vector computer, it is very difficult to keep the routing length within 2 meters. It is even hard to realize in the interface part of the control component. For this reason, when determining routing length the interface routing should be clearly defined and if routing is greater than a certain length or greater than the routing of a certain number of logic levels, then it is necessary to adopt isolation measures. And after isolation, redesign the logic relations of the interface.

3. Redetermining the number of logic levels and the routing length when the above two questions have been thoroughly considered is also difficult, thus when designing a certain degree of overmeasure should be provided. On the basis of the situation involving the several computers in Table 1, it is best if the overmeasure is above 20 percent.

In summary, pulse system and logic chain length is a system index for designing a computer and detailed and comprehensive proofs should be carried out in the design stage with regard to the number of components, packing density and logic divisions and a certain overmeasure be reserved.

This article was written under the guidance of Comrade Li Shuyi [2621 2885 6318] to whom we express our heartfelt thanks.

8226/12712
CSO: 4008/67

FAULT DIAGNOSTIC PROGRAMMING FOR ALU IN THE 757 VECTOR COMPUTER

Beijing JISUAN YANJIU YU FAZHAN [COMPUTER RESEARCH AND DEVELOPMENT] in Chinese Vol 21 No 10, 1984 pp 12-18

[Article by Liao Fujiu [1675 4395 0046], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] I. Introduction

Fault diagnostics for the arithmetic logic unit (ALU) of the 757 computer is realized by software that requires certain hardware support, peripheral diagnostic devices, and methods for detecting and isolating malfunctions in extremely small regions. A fault-location program is used to search a fault dictionary for malfunctions in order to locate and report the source of the trouble down to the primary circuit board level. The development of data structures and fault dictionaries and methods for automatic retrieval have been investigated in actual computer installations and useful results have been obtained. No human intervention is required anywhere in the diagnostic process, and the implementation of automatic diagnostic methods greatly enhances the usefulness of computing machinery and facilitates maintenance.

The following three areas are of primary importance in the software implementation of fault diagnostics for ALU's: 1) selecting the test code; 2) writing the fault diagnostics program; 3) compiling a fault dictionary.

Many other tasks are also involved in developing a fault diagnostic program. The most important of these is to organize the structure of the test data and write software to control the test process, handle the fanning in and out of messages, and automatically search through the fault dictionary and report the diagnostic results while the automatic tests are in progress. The layout and structure of the fault dictionary should facilitate adding to the dictionary and retrieving information from it in accordance with the instructions in the diagnostic program.

Although much work has been published in China and abroad on test code generation for fault diagnostics, few results have been reported regarding the diagnostic programming or fault dictionaries, both of which are indispensable parts of the diagnostic system. This paper is concerned with introducing the operating principles involved in the fault diagnostic program (FDPM) and

fault-dictionary (FDIC) of the arithmetic logic unit. Implementation techniques and various data structures are discussed.

II. ALU Fault Diagnostic Program

The FDPM program is loaded into the system when a hardware failure occurs in the ALU, as evidenced by a repeated failure in performing an operation. In order to simplify the diagnostic procedure, the ALU is divided into certain logic blocks according to structure and the test is carried out block by block. The test codes are fanned into each logic block by calling in the test routine resident in the FDPM that load peripheral devices through an interface. The test program also records the response of the logic block to the test code (the contents of this response are called the received value) and stores it in the diagnostic peripheral device in a Test Results file. After the entire logic block has been completely tested, the test program collates the received responses. If an error is found (the response differs from the expected value) an ALU malfunction is indicated. At this point, the fault dictionary is automatically accessed and the fault locating routine contained in the FDPM examines each entry in the fault dictionary and the corresponding diagnostic message. When the correct entry has been found, the cause of the malfunction and the faulty circuit board are displayed on a terminal or printed out for servicing by technical personnel. Figure 1 shows a block diagram of the FDPM diagnostic procedure.

[Figure 1 on following page]

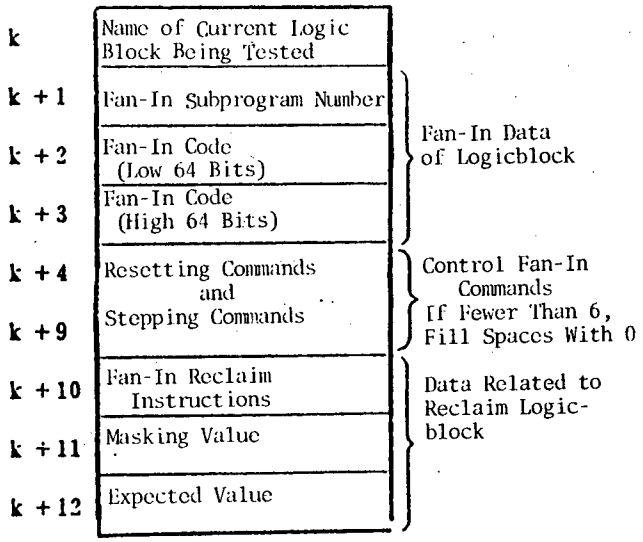


Figure 2 Test Scheme

III. Fanning-In the Test Data and Collating the Responses

1. Test patterns: In addition to the actual test codes, additional data bits must be used to accommodate the input terminal (or register) which sends the

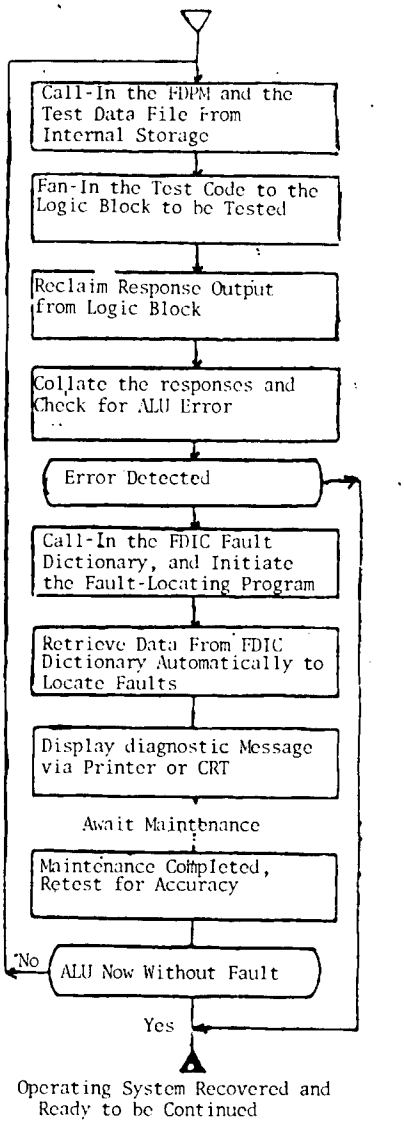


Figure 1 Flowchart for the ALU Fault Diagnostic Procedure

test codes to the logic block; reset instructions and instructions for stepping through the code are also necessary. In order to include these data and instructions in a single test pattern, one must also incorporate the retrieval instructions and the correct responses (expected values) for logic blocks which are not malfunctioning. Figure 2 shows the corresponding fixed data structures for the interrelated test and retrieval data in the test pattern.

This data format facilitates the grouping of related test information together and is a convenient method for organizing large amounts of test data. It also avoids omissions and syntax errors and facilitates programmed examination of the data.

A single text pattern corresponds to carrying out a single test measurement on the ALU. Many measurements and several test patterns are needed to trouble-shoot a single logic block, since each block contains from three or four to several dozen devices requiring different test patterns. The internally resident test patterns are handled continuously in the order in which they are called up. When the diagnostic program is started, each test pattern is entered into a directory which is headed by a single 64-bit word with the format shown in Figure 3. The highest 18 bits give the address of the header; this information is used for partial testing during debugging operations or when the computer is off-line.

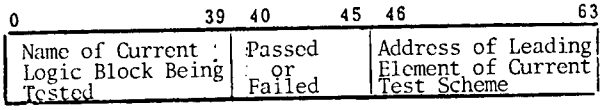


Figure 3 Testing the Directory Character Format

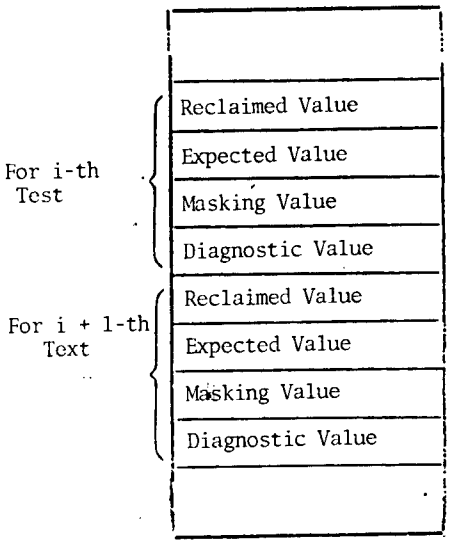


Figure 4 Test Results by Regional Structures

2. Collating the test results: In order to facilitate examination during the subsequent troubleshooting, the FDPM program also arranges related test data in the same order in which the test measurements are carried out. After each test the results are stored in the four information words shown in Figure 4 (reclaimed value, expected value, masking value, and diagnostic value). The

expected and the masking values are initially contained in corresponding test patterns. The masking value is to point out all relevant word places in the current tests (the place corresponds to 1 in the effective word place); the diagnostic value is formed as follows, i.e., diagnostic value = (reclaimed value $\oplus$ expected value) $\wedge$ masking value. All places of the diagnostic value are called diagnostiv places. Fault in ALU exists if the diagnostic value does not equal to 0. A 1 bit in the diagnostic result denotes a discrepancy in the corresponding bit for the reclaimed and expected values and indicates an error at this bit position.

The testing and reclaiming operations are performed on whole words; as a result, the test efficiency is improved because the ALU can handle multi-word bits concurrently during the test procedure.

3. Detecting the presence of a fault: Two methods can be used to decide whether or not an ALU malfunction has occurred--one can attempt to reach a decision after each test measurement has been completed on every logic block, or, one can wait until all the tests on the logic block have been performed and reach a decision on a block-by-block basis. If the first method is used, it is not necessary to continue testing the remaining components in the logic block once a fault has been located; this procedure is thus quicker, but the controlling software is more complicated. If the second method is employed, roughly the same amount of testing time will be required regardless of when the fault is detected; however, the software implementation is simpler. The FDPM program employs the second method, which may be described in more detail as follows.

After all the tests have been completed for a given logic block, the test result memory locations are accessed sequentially to analyze the diagnostic words. If all the bits in a diagnostic word are equal to zero, the program writes all zeros into the directory indicating whether it was pass or failure in the entry corresponding to the current test. These bits indicate the test has been passed. If not all of the locations in the diagnostic value are not equal to zero, the program sets bits all 1's, indicating an ALU failure. In this case the diagnostic program branches to a subroutine which consults the fault dictionary--the fault location program accesses and retrieves data from the fault dictionary--until the problem is identified.

IV. The Fault Dictionary

1. Generating the fault dictionary: As an illustration, consider the logic block on circuit board A101, whose circuit is shown in Figure 5. The signals along the gate K and the code input lines b_1, b_2, b_3 all originate from different circuit boards, and the code output lines c_1, c_2, c_3 also go to different boards. Table 1 illustrates the complete fault-testing and locating procedure for detecting a single malfunction in this logic block (we assume that b_1~b_3, i.e., their signal levels are either both 0's or both 1's).

Table 1 Complete Procedure for Fault Testing and Locating for the Logic Block in Fig. 5

(1)	(2)		(3)	Detectable Fault					
				k		b		c	
	k	b	c	sa 0	sa 1	sa 0	sa 1	sa 0	sa 1
1	1	1	1	1		1		1	
2	1	0	0				1		1
3	0	1	0		1				1

(1) Test Number; (2) Value from Input Terminal;
(3) Accurate Output Value

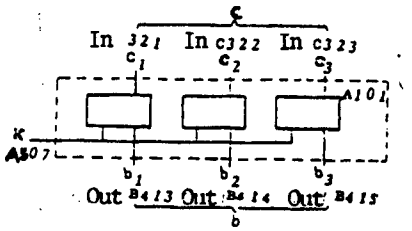


Figure 5 An Illustration of the Logic Block Circuit

Table 2 Fault Dictionary for Testing the Logic Block in Fig. 5

Entry #	Source of Fault		Test # Succeeded	Test # Failed	Possible Faulty Circuit Boards
	Logical Name	sa0/sa1			
1	k, b, c	sa 0		1	A101, A507, C321, C322, C323, B413, B414, B415,
2	c	sa 1		2,3	A101, C321, C322, C323
3	k	sa 1	2	3	A101, A507
4	b	sa 1	3	2	A101, B413, B414, B415

Table 2 shows that several different error sources c, k, and b generate the same signal sa0, i.e., their test responses are identical. In this case, all of the boards from or to which b_1~b_3, c_1~c_3 and k receive or send signals must be suspected if a fault is detected. If there are many bits (lines) involving different circuit boards, the list of circuit boards may be quite long.

Examining Figure 5 reveals that a gate failure will generally result in multiple bit errors; on the other hand, if one of the code bits is wrong, only one bit in the output will generally be in error. This difference is helpful in distinguishing and more precisely identifying these two kinds of faults from their different characteristic bit patterns. Of course, in order to do this one must increase the number of entries in the dictionary. Table 3 shows how the fault dictionary in Table 2 can be enlarged to analyze the incorrect bit patterns.

After the test program has detected a fault, the malfunctioning logic block has been identified, and the results are available for all of the failed tests, the next step is to consult the fault dictionary to identify the failed circuit board.

2. Structure of the FDIC fault dictionary: We see from the above discussion that the fault dictionary basically describes the relationships between a faulty circuit board and the responses of a malfunctioning component in the corresponding logic block to the set of diagnostic tests. In general, the overall response of a malfunctioning component to the battery of tests can be characterized in terms of which tests were passed and which were failed.

Each logic block corresponds to a section (directory) in the dictionary which has the same name as the corresponding logic block and contains several entries, each consisting of a number of items. During automatic data retrieval from the dictionary, each item in each entry must be represented symbolically in conformance with certain structural requirements in order to facilitate programmed retrieval. The fault dictionary is usually quite a large file which is stored in an external memory device during the testing stage and loaded into core only when it must be accessed for retrieval.

Table 3 Expanded Fault Dictionary for Detailed Error Analysis of the Tested Board in Fig. 5

Entry #	Source of Fault			Number of Test Passed	Number of Test Failed	Position of Diagnostic Value = 1	Possible Faulty Circuit Board
	logical name	sa0/sa1	group/bit error				
1	k	sa 0	group		1	1,2,3	A101, A507
2	k, b	sa 0	bit		1	1	A101, B413
						2	A101, B414
						3	A101, B415
3	c	sa 0	bit		1	1	A101, C321
						2	A101, C322
						3	A101, C323
4	k	sa 1	group	2	3	1,2,3	A101, A507
5	k	sa 1	bit	2	3	1 or 2 or 3	A101,
6	c	sa 1	bit		2,3	1	A101, C321
						2	A101, C322
						3	A101, C323
7	b	sa 1	bit	3	2	1	A101, B413
						2	A101, B414
						3	A101, B415

Table 4 Representations and Names of all the Terms in the FDIC Fault Dictionary

Item #	Item Name	Representation	
		Prefix Symbol	Rest of Entry Following the Prefix
1	dictionary section names and entry numbers	F	<section name><entry number>
2	fault source	↑ ..... ↑	<logical name> <sa 0/sa 1> <bit/group>
3	failure condition in current logic block	C [1]	(<condition number>)
4	test to be passed	Y [2]	(<test number>, <test number>, ..., <test number>)
5	test to be failed	N	(<test number>, <test number>, ..., <test number>)
6	test for locating fault	T	(<test number>)
7	register for holding responses	R	(<register name>)
8	bit error	E [3]	(0)/(1)
9	diagnostic bit		
10	faulty circuit board	:	<board number> <board number> ... <board number>)

[1] The second symbol ↑ is a terminator;

[2] Some faults may be indicated using Y, N terms or just an N term alone; in this case the C or the C, Y terms may be omitted;

[3] This term is omitted for multiple bit errors in which some of the wrong bits are 1's and some are 0's.

Various prefixes and tags are added to the entry items in the FDIC fault dictionary in order to assist the automatic retrieval process. Table 4 lists the names and representations of each of ten items, the maximum number that any entry can contain. An entry can be formed by stringing names together in accordance with the syntax in Table 4. For example,

FYWWC 1↑K←0 SA 0 SGL↑C(37)N(115, 210)T(115)R(E)E(1)
P 0/P 60(4):B 114, P 1/P61(4):B 124, P 2/P 63(4):C 103, P 3/P 64(4):C 108.

Since all the items in the FDIC are represented using ASCII coded characters, the entire FDIC fault dictionary is an ASCII file.

3. Representation of the diagnostic bits: The ALU has a 64-bit word length and the digital code is carried by lines which originate and terminate in numerous circuit boards; when analyzing wrong bits, one therefore attempts to localize the source of the trouble and pinpoint the failure more precisely by minimizing the number of boards that must be investigated. It is therefore not enough for the fault source items in the FDIC to merely list the logical name of the malfunctioning component and the nature of the failure--a quick method is also required for exhibiting the relationships between the partial bits in the diagnostic word and the circuit board which is malfunctioning.

The FDIC uses the following types of diagnostic expressions.

1) P-expressions: In this case one wants to check if a certain diagnostic bit is equal to 1. This diagnostic bit is distinguished by the prefix symbol P. The two types of relations, "and" and "or," between the diagnostic bits may be described as follows:

① The "or" relation is denoted by an expression of the form Pi_1, Pi_2, ..., Pi_n and is satisfied if and only if at least one of the bits i_1, i_2, ..., i_n is equal to 1.

② The "and" relation is denoted by $Pi_1Pi_2...Pi_n$. This relation is satisfied if and only if each of the bits i_1, i_2, ..., i_n is equal to 1.

③ The syntax for the "or" relation can be simplified to Pi/Pj(k), where k is the step length. This relation is satisifed if and only if at least one of the bits i, i+k, i+2k, ..., j is equal to 1.

④ The "and" relation can also be expressed in the simple form Pi-j(k), which is satisifed if and only if all of the bits i, i+k, i+2k, ..., j are equal to 1.

Expressions of the form ① and ③, ② and ④ can be combined; e.g., P8/P15, P22, P24, and P3-7, P11, respectively.

⑤ There is also an "and or" relation which has the form <and relation>/<and relation>/...<and relation>. The separating slashes "/" indicate that the

various "and" expressions are to be "or"-ed. That is, the entire relation is satisfied if and only if one of the "and" relations in the chain holds.

2) Q-relations: These are used to test if a specific diagnostic bit is equal to zero; these bits are denoted by the prefix Q. The form of the Q-expressions is completely analogous to the form of the P-expressions, from which they can be obtained simply by replacing P by Q. For example, $Qi_1Qi_2...Qi_n$ holds if and only if all of the diagnostic bits i_1, i_2, ..., i_n are equal to zero.

3) @-relations: These are formed by adding the symbol @ in front of a P-relation; e.g., @P1P3 or @P0/P63(4).

The meaning of an @-expression is similar to that of the corresponding P-expression, except that in order for an @-expression to be satisfied it is not enough that the corresponding P-expression hold (i.e., that the relevant diagnostic bits be equal to 1)--in addition, all of the other diagnostic bits not specified in the relation must be equal to zero.

The items specifying the diagnostic bits are paired with the items which characterize the faulty circuit board in each entry in the FDIC fault dictionary. The pairing can be iterated and the elements in a pair are separated by a comma ","; the last pair in the sequence is followed by a semicolon";" which serves as a terminator.

V. Retrieval From the Fault Dictionary and Location of the Malfunction

Figure 6 illustrates the retrieval and location procedure.

1. Make a subdirectory: A subdirectory heading is provided for each section of the dictionary file using the format shown in Figure 7.

2. Search for failed tests: The fields indicating whether a test was passed or failed are scanned within a directory. The scanning process terminates when a field with all 1's is encountered, and the name (NAME) of the failed logic block for this test is read from the directory.

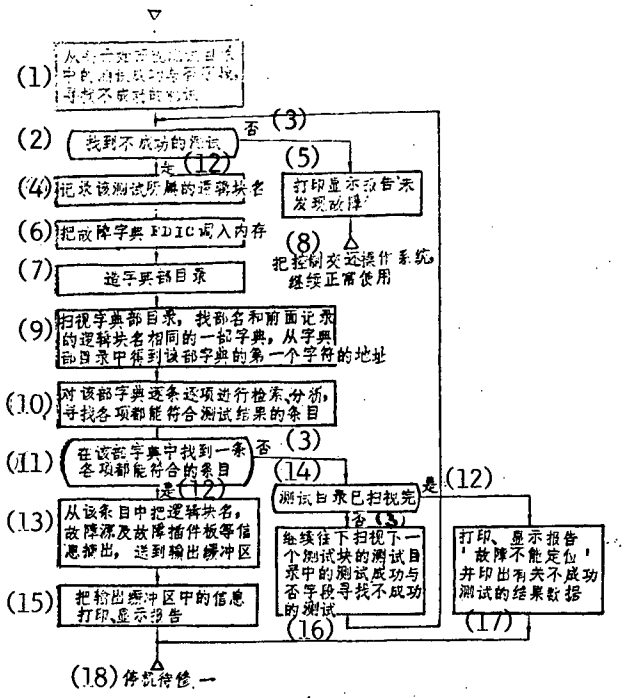


Figure 6. Block Diagram Illustrating Retrieval From Fault Dictionary and Dictionary Locations

Key:

1. Scan all the test passed/test failed fields in the directory and look for failed tests
2. Failure found
3. No
4. Record name of logic block which failed test
5. Display message "no error found"
6. Call in the FDIC fault dictionary
7. Create a dictionary directory
8. Return control to the operating system and continue normal operation
9. Search the dictionary directories for the section whose name and prefix symbol match with the logic block name, and retrieve the address of the first symbol from that directory
10. Search through the entries and items of this directory to see if any correspond to the test results
11. Was a match found?
12. Yes
13. Extract the name of the logic block, source of malfunction, faulty board, and other information from the matching entry and send it to an output buffer
14. Search of test directory complete?
15. Display information in output buffer on terminal screen and print it out
16. Continue searching through the test failed/test passed fields for the next logic block in the test directory and look for failed tests
17. Print out and display message "Unable to locate failure" and print out the relevant data for the tests that were failed
18. Stop the computer for repairs

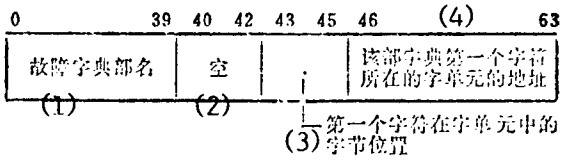


Figure 7. Directory Character Format in the Fault Dictionary

Key:

1. Section heading
2. Unused
3. The byte position for the first character in the word cell
4. Address for the word cell containing the first character for the current dictionary section

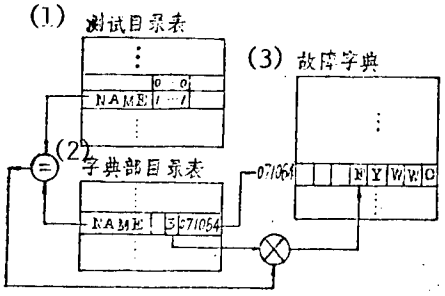


Figure 8. Procedure for Searching for the Location of the Leading Characters in the Fault Dictionary

Key:

1. Table of test directory
2. Table of dictionary section directory
3. Fault dictionary

3. Find the dictionary section to be examined: Each logic block corresponds to a section of the dictionary with the same name. In order to speed up the dictionary search, the fault dictionary is called in only when tests have been failed, and only the section with the same name as the failed logic block is searched. The schematic in Figure 8 shows how the leading character in the section heading is located.

4. Examine the dictionary and locate the fault: All the entries in the relevant section of the fault dictionary are examined by inspecting each item for a match with the test results. If a matching entry is found, the dictionary search terminates successfully and the malfunction has been located; the source of the trouble and the faulty board can then be reported using information in the matching entry. Figure 9 illustrates how the dictionary is examined.

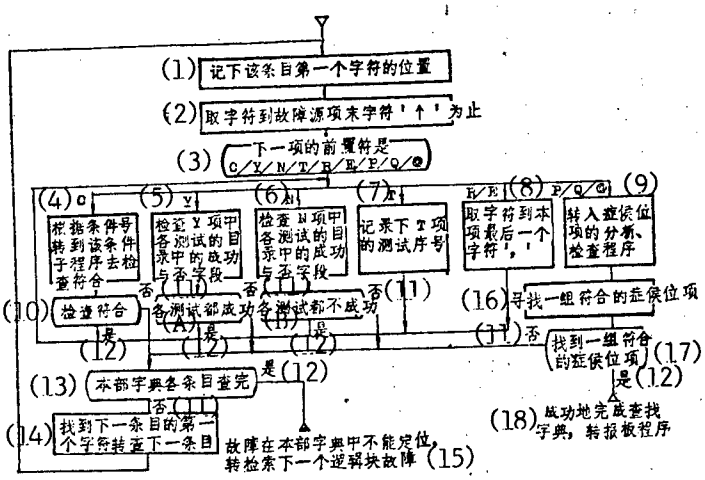


Figure 9. Flow Chart Illustrating the Use of the FDIC Dictionary

Key:

1. Record location of first character of current item
2. Read characters in the fault source item until the terminator ↑ is reached
3. Prefix of next item is ...
4. Use the condition number to execute the corresponding subroutine and look for a match
5. Examine all the test passed/test failed fields in the Y item
6. Examine all the test passed/test failed fields in the N item
7. Record the test numbers in the T item
8. Read all characters in the current term up to the last symbol ","
9. Execute program for examining and analyzing the diagnostic bits
10. Examine characters
11. No
    A) All tests passed?
12. Yes
    B) All tests failed?
13. All items in current dictionary section examined?
14. Find first character in next entry and continue examining
15. Unable to identify fault within current dictionary section; search to see if the next logic block is at fault
16. Search for a group of corresponding diagnostic bits
17. Corresponding diagnostic bits found?
18. Dictionary search successful, execute program to report the faulty board

Service technicians must examine the fault source, the E, and the R terms in each item contained in the dictionary entry during the repair stage; the C, Y, and N items and the diagnostic bits must be checked one-by-one to see if they correspond; the test numbers contained in the T item are used when the diagnostic bits are examined. If examination of any of the C, Y, and N terms reveals a mismatch, the entire entry fails to match the test results and the

next entry is consulted. If these three items all match, the groups of diagnostic bits in the diagnostic item are inspected; if one of these groups matches the test result, the entire entry matches and the fault has been successfully located. The number of the faulty board is present in the "faulty board" item which is paired to the groups of matching diagnostic bits. This process is illustrated by Figure 10.

It should be noted that the sequence of diagnostic bits in the diagnostic word is determined by the masking word (ie., by the mask bits which are equal to 1). For example:

masking value: 00001111110...011111
diagnostic value: 10000010000...000000
diagnostic bit positions: 012345 678910

Thus, only bit 2 in the diagnostic field is equal to 1.

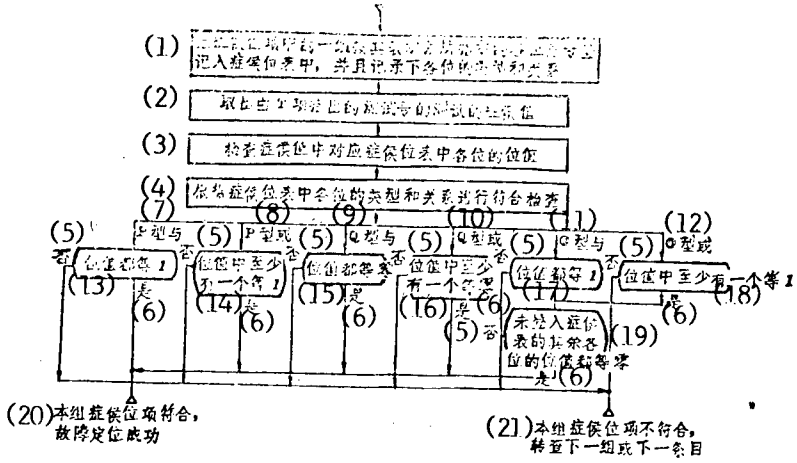


Figure 10. Flow Chart for Examining Correspondences in the Diagnostic Bit Field

Key:

1. The diagnostic field contains a group of bits which is specified by the diagnostic expression (cf. Sec. 4, part 3). Record these bits in the diagnostic bit table, together with the type of each bit and the relation containing them.
2. Get the diagnostic fields for the tests listed in the T item.
3. Check the bit positions of the bits in the diagnostic field which match the ones in the diagnostic bit table.
4. Check to see that the type and relation for each of these bits matches the corresponding data in the diagnostic bit table
5. No
6. Yes

[Key continued on following page]

[Key for Figure 10 continued)

7. "and" P-type
8. "or" P-type
9. "and" Q-type
10. "or" Q-type
11. "and" @-type
12. "or" @-type
13. All bits 1
14. At least one bit is 1
15. All bits 0
16. At least one of the bits is 0
17. All bits 1
18. At least one bit is 1
19. All of the remaining bits (not entered in the diagnostic table) are equal to 0
20. This group of diagnostic bits matches, fault has been located
21. This group of diagnostic bits does not match; check next group of bits or next entry

5. Error reporting: When a matching group of diagnostic bits has been found, the information from the corresponding Faulty Board item (i.e., name of the logic block) is read and output to a printer or terminal, where it is analyzed by repair technicians. If necessary, the technicians can also instruct the computer to print out the data on the failed tests. In those instances when a fault has not been located, this information may be helpful for human analysis of the cause.

VI. Repairing the System

After the technicians have repaired the malfunction, the system can be retested. If no failures occur during continuous operation, control is transferred to the operating system and normal operation resumes (it is not necessary to perform a cold restart).

VII. Conclusions

The features of the fault diagnostic program and fault dictionary developed for the ALU of the 757 computer may be summarized as follows.

a) A field (pattern format is used to represent the test data and other information; b) data retrieval from the dictionary and fault location are programmed, and the entire fault diagnostic process is highly automated; c) programmed retrieval is facilitated by using distinctive prefixes in all of the items in the fault dictionary; d) the diagnostic efficiency is increased because full data words are examined concurrently; furthermore, measures are taken to limit the range of possible fault locations during the stage when the wrong bits are analyzed; e) the facilities available for checking syntax and reporting errors can handle large amounts of data.

Because the system is new, the FDIC fault dictionary still has a few shortcomings, of which the most serious are:

1) Although the accuracy in locating fault by analyzing wrong bits is increased, this is at the cost of considerably increasing the number of entries in the fault dictionary. If a more concise method could be found for expressing the relevant information, fewer entries would be needed and the size of the dictionary could be reduced. 2) The diagnostic syntax was developed by a process of ad hoc additions made at various times; as a result, it is not always as simple and clear as it should be.

We have tested the FDPM fault diagnostic program for some time and it has performed as expected. Approximately 10 seconds is required to output the diagnostic report after a failure has been detected, regardless of whether the trouble can be quickly repaired or not.

Xu Jun [1776 3182], Sheng Chuanying [4141 0278 5391], Zhang Hua [7022 5478], Zheng Shuhua [6774 3219 5478], and several other people participated in the writing of this article.

12,617/9599
CSO: 4008/146

MAGIC ORDER MERGE-SORT ALGORITHM

Beijing JISUANJI XUEBAO [CHINESE JOURNAL OF COMPUTERS] in Chinese Vol 7 No 5, Sep 84 pp 353-358

[Article by Zheng Zhijie [6774 2535 2212], Institute of Computing Technology, Chinese Academy of Sciences]

[Text] Abstract. A magic order merge-sort algorithm is introduced. This algorithm has a simple regulated control form, an overall characteristics of harmonious symmetry, and has higher application values.

The derivation of an algorithm is given from a biquadratic linked merge algorithm: the magic merge-sort algorithm. Since the distance between comparands in each pass of the execution of this algorithm form an array whose elements are such that, according to the Oriental mathematical tradition, they form an unusual set with combinatorial meaning, called the magic numbers. The author borrows this term to call it "the ought to algorithm," to emphasize the important effect of describing the relationship between the distance value and place value within the algorithm. It should be pointed out that this is not a new algorithm, and that its external form is the same as Batcher's odd-even sort algorithm given in reference [1]. However, this is independently derived by the author from another viewpoint. The algorithm utilizes the control form of the distance value and numerical place value to fundamentally change the view on the algorithmic function, thereby overcoming the deficiency of large operational values of the algorithm in reference [1].

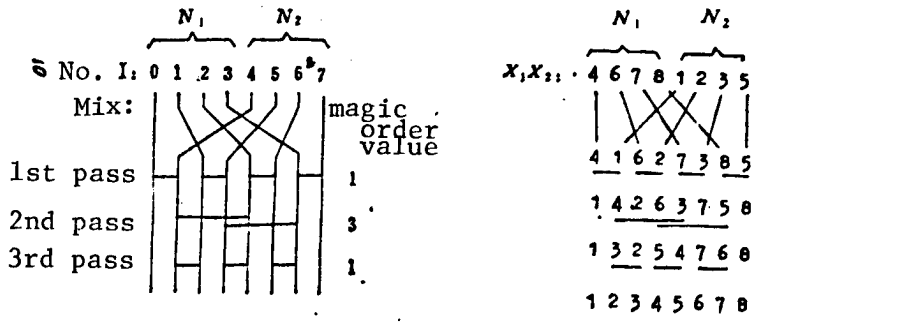
I. The Principle of the Algorithm

The basis of a merge-sort algorithm is merging. The sorting is formed by repeated mergings. Figures 1 and 2, respectively, give a scheme and an example of a merging and sorting problem, and they are given to show the algorithmic process. One needs to carry on a mixed preprocessing while merging two ordered arrays, but this process is not required in sorting.

Paper received 12 May 1982.

In the following descriptive algorithm, I is the serial number

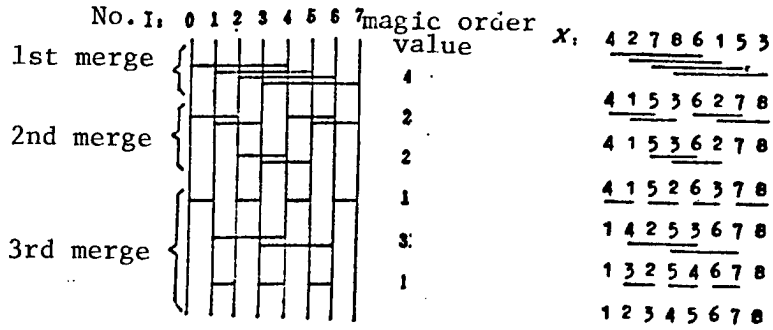
$$I = \sum_{j=0}^{n-1} I(j) \cdot 2^j,\ 0 \leqslant I < N,\ n = \lceil \log N \rceil,\ I(j) \in \{0, 1\}.$$



(a) (4,4) merge scheme (b) a (4,4) merge example

Figure 1 (4, 4) merge

In (b) $x_1[0:3] = (4, 6, 7, 8)$, $x_2[0:3] = (1, 2, 3, 5)$



(a) 8 element sort scheme (b) a sort example

Figure 2 Magic order merge sort, N=8

In (b) $x = x[0:7] = (4, 2, 7, 8, 6, 1, 5, 3)$

Method: Magic order merge sort algorithm

Input : $X[0:N-1]$;

Output: $Y[0:N-1]$, X 's one replacement

$Y[i] \leqslant Y[i+1]$, $0 \leqslant i \leqslant N-2$;

Process: Run magic merge-sort procedure MOS[N]. The algorithm is formed by two nested loops. The inner loop executes the completion of merge in each pass and the outer loop ascertains the processing of each merge. The elements to be compared (comparands) are selected in concert with the merge number, pass number, and array serial number, while exiting the outer loop the array elements are arranged in sequence by their magnitudes.

Compare/exchange procedure: CI[I,J,B,H]

```
procedure CI[I, J, B, H]
  do
    if I[J] = B then do
      if X[I] > X[I + H] then do
        X[I] ⟷ X[I + H];
      end;
    end;
end;
```

Magic order merge-sort procedure: MOS[N]

```
procedure MOS[N]
  do
      m ← ⌈log N⌉ − 1;
      j ← m;
      while j ≥ 0 do
        h ← 2^j;
        for 0 ≤ I < N − h do
        CI[I, j, o, h];
        end;
        k ← m;
        while k > j do
          h ← 2^k − 2^j;
          for 0 ≤ I < N − h do
            CI[I, j, 1, h];
          end;
          k ← k − 1;
        end;
        j ← j − 1;
      end;
end.
```

II. The Precision of the Algorithm

As the external characteristics are guaranteed in [1], we give a simple explanation of the relationship between this algorithm and the biquadratic numbers linked merge algorithm. Let these be two 2^{n-1} cascade connected sorted data arrays, respectively merged, one using the biquadratic linked merge algorithm and the other by magic order merge algorithm. Let the date array using the former algorithm be designated X, and the latter by $\mathscr{X}$,

then the elements of the two arrays in comparison/exchange after each pass may have a corresponding relationship as shown in Figure 3.

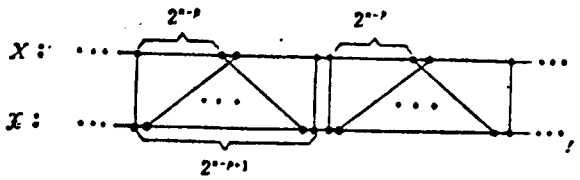


Figure 3. The Corresponding Relationship of the Two Data Arrays After the P-th Pass

The figure shows that after the P-th pass processing, the elements on $\mathscr{X}$, divided into a series of segments of length (2^{n-p+1}), have a one-to-one correspondence with the 2^{n-p} elements in the similar distance under the mixed interpretation.

When the two merged data arrays have different lengths, i.e., $N_1 \geqslant N_2$, then the magic order merge format requires it to add to the second data array elements such that N_2 will be equal to N_1 or $N_1 - 1$.

The sorting is achieved by repeatedly merging subarrays, and after merging, the subarrays are still interleaved in $\mathscr{X}$. The number of the sorted subarrays will decrease by one-half after each merge. After n merges on the arrays, the array elements will be sequentially arranged. Consequently, from the fact that it is not necessary to augment the initialized operation, it can be said that the processing format is unified.

III. Parametric Analysis

Comparison frequency C[N]

Let C[j, k, N] be the N elements, at the k-th merge, the j-th pass processing comparison frequency number is

$$C[N] = \sum_{k=1}^{n} \sum_{j=1}^{k} C[j, k, N] \tag{1}$$

as the first pass differs from the remaining passes, then

$$C[N] = \sum_{k=1}^{k} C[1, k, N] + \sum_{k=2}^{n} \sum_{j=2}^{k} C[j, k, N]$$

In separately discussing the two values, for convenience, we use the following designations:

$$N(k) = N(\text{mod } 2^{n-k});$$
$$N(j, k) = (N - h(j, k)) \ (\text{mod } 2^{n-k});$$

where h(j, k) is the magic order value of the k-th merge and j-th pass processing, $h(1, k) = 2^{n-k}$, $h(j,k) = 2^{n-j+1} - 2^{n-k}$, $j \geqslant 2$

As

$$\delta_k = \left\lfloor \frac{N}{2^{n-k}} \right\rfloor (\text{mod } 2);$$

$$\delta_{j,k} = \left\lfloor \frac{N - h(j, k)}{2^{n-k}} \right\rfloor (\text{mod } 2);$$

then

$$C[1, k, N] = \left\lfloor \frac{N}{2^{n-k+1}} \right\rfloor \cdot 2^{n-k} + \delta_k \cdot N(k);$$

$$C[j, k, N] = \left\lfloor \frac{N - h(j, k)}{2^{n-k+1}} \right\rfloor \cdot 2^{n-k} + \delta_{j,k} \cdot N(j, k), \quad j \geqslant 2;$$

The parameter value can be obtained by substituting the above into (1).

In particular, when N = 2, N(k) = N(j, k) = 0, and

$$\begin{aligned} C[2^n] &= \sum_{k=1}^{n} \left\lfloor \frac{2^n}{2^{n-k+1}} \right\rfloor \cdot 2^{n-k} + \sum_{k=2}^{n} \sum_{j=2}^{k} \left\lfloor \frac{2^n - 2^{n-j+1} + 2^{n-k}}{2^{n-k+1}} \right\rfloor \cdot 2^{n-k} \\ &= \sum_{k=1}^{n} 2^{n-1} + \sum_{k=2}^{n} \sum_{j=2}^{k} (2^{n-1} - 2^{n-j}) \\ &= 2^{n-2} \cdot n \cdot (n-1) + 2^n - 1. \end{aligned}$$

Comparison pass number $C_p[N]$

$$C_p[N] = \sum_{k=1}^{n} \sum_{j=1}^{k} 1 = \binom{n+1}{2} \tag{2}$$

Exchange frequency number I[N]

For each comparison, there is an accompanying exchange, so

$$I[N] \leqslant C[N] \tag{3}$$

Exchange pass number $I_p[N]$

$$I_p[N] = \binom{n+1}{2} \tag{4}$$

The different distance interval number DIN[N]

In order to determine its value, the characteristics of magic order is examined

$$\begin{cases} h(1, k) = 2^{n-k}, & 1 \leqslant k \leqslant n; \\ h(j, k) = 2^{n-j+1} - 2^{n-k}, & 2 \leqslant j \leqslant k \leqslant n; \end{cases}$$

Clearly, besides $h(1, k) = h(k, k)$, $2 \leqslant k \leqslant n$, the other values are all different, therefore,

$$DIN = \sum_{k=1}^{n} \sum_{j=1}^{k} 1 - \sum_{k=2}^{n} 1 = \binom{n}{2} + 1 \tag{5}$$

Transmission frequency DT[N]

$$DT[N] = C[N] + I[N] \tag{6}$$

Transmission pass number $DT_p[N]$

As the parameters and the support operation are related to the system structure, we utilize an equidistant interconnected model

$$DT_p[N] = \sum_{k=1}^{n} \sum_{j=1}^{k} 2 \cdot TS[j, k].$$

where TS[j, k] is the transmission pass number, necessary distance to transmit h(j, k). If the interconnected distance $\{2^i\}_{i=0}^{n-1}$, except for h(1, k) can reach after the first pass, the others need two passes.

$$DT_p[N] = \sum_{k=1}^{n} 2 + \sum_{k=2}^{n} 3 + \sum_{k=3}^{n} \sum_{j=3}^{k} 4 = 2(n^2 - n + 1) \tag{7}$$

If the interconnected distance $\{h(j, k)\}_{0 \leqslant j < k \leqslant n}^{1 \leqslant k \leqslant n}$, then TS[j, k] = 1

$$DT_p[N] = \sum_{k=1}^{n} \sum_{j=1}^{k} 2 = n(n + 1) \tag{8}$$

The influence of interconnected form on the computational speed can be seen from these paramters.

Auxiliary storage volume AM[N]

$$AM[N] = 0 \tag{9}$$

Auxiliary operation frequency AON[N]

Auxiliary computations: (1) computing h; (2) N-H; (3) 1+H; (4) k-1. (1), (2), and (4) need to be computed once, and (3) needs to be computed as many times as the comparisons. Since these are all arithmetic operations, then

$$AON[N] = \sum_{k=1}^{n} \sum_{j=1}^{k} 3 + C[N] = 3\binom{n+1}{2} + C[N] \tag{10}$$

Since the controls require only logic decisions to form the sorting, then the complexities can be neglected.

Auxiliary operation pass number $AON_p[N]$

The processing of I+H can be concurrently done

$$AON_p[N] = 4 \cdot \binom{n+1}{2} \tag{11}$$

the operation time analysis is similar to that of (2).

The author has performed tens of millions of simulated computations on the 757 vector machine, and has actually carried out operations of this algorithm, through using sequence value and place value for description, and from this angle, successful reduction control complexities have been achieved.

This paper is part of the author's thesis, and he wishes to express his gratitude to Professors Gao Qinshi, Zheng Xiang, and Zhou Xaobe for guidance and assistance.

REFERENCES

1. Knuth, D.E., "The Art of Computer Programming," Vol 3 (Sorting and Searching), Addison-Wesley, Reading, MA, 1973, pp 111-113.

2. Lorin, H., "Sorting and Sort Systems," Addison-Wesley, Reading, MA, 1975, pp 358-365.

3. Zheng Zhijie, "The Duodirun Merging Algorithms (1) and (2), JISUANJI XUEBAO [CHINESE JOURNAL OF COMPUTERS], 6:2, 4 (1983).

12744/9365
CSO: 4008/101

LIST OF AUTHORS (with the number of each article indicated in parentheses):

/6539
CSO: 4008/101

END