Consiglio Nazionale delle Ricerche
ISTITUTO DI ELABORAZIONE DELL'INFORMAZIONE

A MULTI-PURPOSE INTERPOLATOR FOR NUMERICALLY CONTROLLED MACHINE TOOLS °

V. Casarosa, G. Frosini, G. B. Gerace,
P. Maestrini, P. Piram

Nota Interna B69-12
(Settembre 1969)

° This work has been supported by C.N.R. - "Programma di ricerca sull'automazione dell'industria meccanica" under contract n° 115/2258/1500.

1. Introduction

In numerically controlled contouring machine tools the machining operations are defined by a suitable set of geometrical and technological data, which have to be conveniently processed in order to achieve the actual tool path. As a final result of data processing, the coordinates of the points to be reached by the tool at discrete time intervals are calculated and transmitted in real time to the machine actuation system, achieving a continuous motion which fulfils speed and acceleration requirements.

Data processing is usually performed as follows. First, geometrical data are processed by means of a general purpose computer, and a simple and concise definition of the tool path is achieved (e.g. consisting of straight segments only). Secondly the computation is completed by means of a special purpose real time computer, called interpolator or director, permanently connected to the machine tool.

The outstanding advantages of this solution consist in not employing the general purpose computer in a great amount of input-output operations. For the on-line computer the special purpose approach is usually preferred since it is more easily operated by unskilled people [1],[2].

On the other hand this solution leads to a great variety of interpolators, in order to fit different applications. Differences arise from the type of machine-tool being controlled (e.g. lathe, milling machine, etc.), from the number of controlled axes, from the geometrical elements of which the tool path consists (e.g. straight segments, circular or parabolic arcs, etc.), from the more or less sophisticated techniques of speed and acceleration controlling.

In this paper an approach to the interpolator design is discussed, that overcomes the disadvantages mentioned above. The interpolator consists of an unconventional small computer, whose restricted instruction set has been chosen for general and efficient use within the whole class of the applications above.

In order to preserve the advantages of the special-purpose computer, the program specializing the interpolator performance is stored in a read-only memory, read-write memory being reserved for data and operands. The multi-purpose capabilities of the computer allow a single main frame to be sufficient for the whole class of applications, the specialization being achieved by the replacement (total or partial) of the read-only memory.

According to this philosophy, a computer is being realized at the "Istituto di Elaborazione della Informazione", Pisa, within the Program for the enhancement of the automation of the mechanical industry, sponsored by Italian National Research Council (C.N.R.).

2. Basic System Design

The layout of the computer, as well as the list and the format of the instructions, are strictly connected to the interpolation algorithms characteristics and to the real time requirements which can be summarized as follows:

a) the number of data and operands stored at the same time is relatively small;

b) most of the operations to be performed are arithmetic; usually multiplications and divisions occur only by powers of ten;

c) the operation flow heavily depends on either the sign or the value (zero or not) taken by the quantities during the computation;

d) a relatively high rate of real time transfers of BCD input and output data is required.

The basic system design that has been determined starting from these characteristics and requirements is now described.

2.1 Arithmetic

In order to fulfil the most severe accuracy requirements occurring in the area of numerical control (a resolution of 1 micron for displacement has been assumed), the length of numerical data has been fixed in 8 decimal digits. To avoid the frequent and intricate conversion of BCD input and output data, the computer has been based on decimal arithmetic and the excess-three code has been used. As required by the applications, the numbers are represented as integers (negative numbers in 10's complement form).

2.2 Memories

Since the interpolation algorithms operate at a given time on a small number of data and operands, the read-write memory (RWM) can be of a very small capacity (a maximum of 64 words has been assumed) and then implemented with very high speed techniques. The program specializing the performance of the interpolator is stored in a wider read-only memory (ROM), with resulting reduced cost and increased realiability. The high speed capabilities of the RWM allow serial execution (one decimal digit at a time) of arithmetic instructions, with consequent reduction of the arithmetic unit circuitry.

2.3 Instruction Format

The great number of arithmetic operations necessary to perform the interpolation algorithms makes it convenient the choice of a three-address instruction format that allows the reduction of the number of the ROM words, together with an easier programming.

This solution results to be a practical one since, owing to the small size of the RWM, only few bits (6 bits) are needed to address an operand. With a three-address instruction format, arithmetic operations such as A+B=C , where A, B and C are stored in the RWM, can be executed by one instruction only. Conditional jumps, which frequently occur in programming the algorithms, can also be executed by a single instruction that contains the address of the next instruction and the address of a quantity conditioning the jump.

18 bits have accordingly been assigned to the address field of an instruction. Thus arithmetic instructions address three RWM words, and jump instructions address by 6 bits a conditioning quantity and by 12 bits the next instruction in the ROM. Such an address field width enables the direct addressing of a ROM with sufficient capacity for the most sophisticated applications in the numerical control area.

The six bits specifying the address of a conditioning quantity can represent either the address of a variable stored in the RWM or the address of an indicator storing a single bit information such as input-output flag, signal from the consolle, internally generated signal, etc.

More sophisticated solutions such as variable length instruction formats, paging techniques for addressing the ROM, conditional jumps implemented by skip instructions, all oriented in reducing the

instruction length, have been found not to lead to a reduction in the total number of required ROM bits.

Since read-only memories with a large word length are readily and economically available,a trend to reduce the total number of words rather than their length has been found reasonable.

2.4 Input-output

Alphanumeric input data defining tool path and machining operations are transferred from low speed external devices (tipically a punched tape reader) into the RWM. Data are supplied as a sequence of words, grouped in blocks, each block referring to an element of the trajectory. A block must be entirely transferred into the RWM before interpolating the corresponding trajectory element. Input operations must not cause the interruption of the tool motion. This is achieved reading in, word by word, the data that refers to an element while interpolating the preceding one. Each input device is thus provided with a buffer whose length need only be a word (8 decimal digits).

Buffers are automatically filled by the corresponding external devices. As soon as a buffer has been completely filled, a flag is lifted and the external device stops itself.

The task of transferring data from buffers to the RWM, resetting flags, starting input devices, recognising and checking data is assigned to the program specializing the performance of the interpolator.

Output data, such as tool-motion and further operation commands, are transferred in real time from the RWM to the digital-analog converters of the servomechanisms. The main program controls

the transfers from the RWM to the input buffers of the various devices.

3. Instruction Set

Due to the limited number of different operations required in interpolation algorithms, 4 bits have proved to be sufficient for the basic operation code of instructions.

According to what was above said, the instruction set nearly consists of arithmetic and jump instructions.

Among the arithmetic instructions, besides those performing addition and subtraction, the set contains the following unsual instructions (written in mnemonic form, where L_i represents a RWM address and (L_i) its contents):

ADS $L_1 L_2 L_3$: add (L_1) to (L_2) or subtract (L_1) from (L_2) according to the sign of contents of a fixed location, and store the result in L_3;

MOD $L_1 L_2 L_3$: multiply the signs of (L_1) and (L_2), and store the result in the sign bit of L_3;

ABS $L_1 L_2$: transfer the absolute value of (L_1) in L_2.

Multiplication and division by powers of ten are executed by digit-shifting instructions.

Jump instructions include unconditional jump, (direct and indirect via the contents of a RWM word) and the following conditional jumps (F represents a ROM address, S an indicator address and (S) its contents):

TZE F L_1 : take the next instruction from ROM address F if

(L_1) equals zero, otherwise execute the next sequential instruction;

TPL F L_1 : the jump occurs if (L_1) is greater than or equal to zero;

TRS F S : the jump occurs if (S) is "1".

Besides the indirect jumps, another instruction links the two distinct memories:

STO C L_1 : store the 12 bits configuration C in the 3 least significative digits of L_1,

where the C configuration can be regarded as a numerical constant for arithmetic operations or as a ROM address for use in indirect jumps. Thus the use of subprograms in coding the interpolator algorithms becomes possible, as the return to the calling point can be made via a RWM word.

Two powerful instructions accomplish input and output operations, respectively. They perform the necessary conversion from external to internal (or viceversa) BCD code and transfer a variable number (from 1 to 8) of digits between a selected input or output channel and an arbitrary RWM word.

The instruction set also contains instructions for transfers between RWM words and for setting or resetting indicators.

4. Computer Design and Implementation

The outstanding features of the multi-purpose interpolator will be summarized next.

The ROM capacity varies with fixed increments of 64 words up to

the maximum of 4096. Word length is 22 bits. Every 64 word section is contained in a plug-in package.

Semiconductor integrated devices have been used to implement the ROM, thus reducing the interface circuitry. In particular, an output register is not required.

The RWM is a random access store, implemented with integrated, TTL 16 bits arrays. Read-out is non-destructive. Read and write times are less than 50 nsec.

The RWM is organized on a 4 bits cell basis, with a maximum storage capacity of 512 cells. Instructions make reference to words, formed by 8 consecutive cells.

The high cost of integrated devices for the RWM is largely justified by the advantages of non-destructive read-out capability, simplification of interface and high speed.

Owing to the features of the ROM and the RWM, which behave respectively like a combinatorial network and like a set of addressable registers, the whole system can be considered as consisting only of combinatorial networks and registers.

Since the computer register elements are all of the master-slave type, all transfers among the registers through combinatorial networks can be performed at the same time, and controlled by a single phase clock, simply activating register elements.

A system with this structure can be easily described by a set of micro-programs, written in a transfer language and methods can be used to translate the system description into logical diagrams.

The special purpose computer described here has been designed following an approach discussed in [3], which also reduces the logical complexity of the final solution.

The computer has been formally described by a set of micro-programs executing the instructions and by a special micro-program fetching the next instruction to be executed.

Each micro-program is formed by a list of micro-instructions and each micro-instruction is formed by a list of groups of transfer relations. Each group of transfer relations consists of a set of register transfer relations (this set may be void), corresponding to elementary operations of the system, and of one transfer relation specifying the jump to the next micro-instruction. Only one group of transfer relations is executed at the same time, the selection depending on the value of conditioning variables.

Owing to the simple structure of the system, micro-instructions used in the description consist of at most two groups containing identical register transfer relations (only the jumps to the next micro-instruction result to be conditional), and the micro-program fetching the next instruction consists of one micro-instruction only (fetch micro-instruction).

Processing the formal description by the algorithms described in [3], [4] leads to partition the system in two interconnected automata. Such automata result completely defined and correspond to the operative portion and to the control unit of the computer.

The general layout of the system is shown in Fig. 1, where heavy connections represent the main information flow while the others refers to addressing.

Besides the RWM, the ROM, internal buses and other circuitry, the operative portion includes an arithmetic unit that has been implemented by a multi-level network. It consists of a 4-bits register, a 4-bits binary adder, switches and flip-flops storing carries and overflows.

The automaton representing the control unit has been implemented by a read-only memory storing micro-orders, each of which correspond to a micro-instruction. This memory has been implemented by the same techniques as the ROM, thus obtaining packages interchangeability.

As a consequence of the simplified form of micro-instructions, the micro-order format is very simple, as shown in Fig. 2. The application of the synthesis formal method results in obtaining the minimum number of commands which control the logical networks of the operation portion. The CV field contains such commands, encoded to obtain a further reduction of word length. The CD field identifies the external variable conditioning the sequence of micro-orders. If the value of the conditioning variable is 0, the address of the next micro-order to be executed is taken from the AD field; otherwise such address is either the next sequential one (B=0) or the address of the micro-order corresponding to the fetch micro-instruction (B=1).

The computer also includes a 6 msec time base clock, which can be sensed and started as an indicator. This facility enables real time computations of speed and acceleration.

5. Final Remarks

In this paper a specialized multi-purpose computer for applications in the area of numerically controlled machine tools has been described.

The computer has been conceived as an attempt to give a rather unified and systematic solution to a class of computational problems commonly occurring in the industrial practice, as opposed to the usual approach of designing a distinct special-purpose

system for each application.

The proposed multi-purpose computer can be specialized for a given application by simply defining a proper program to be stored in a ROM. This is an easy task, as programming aids (such as a simulator and an assembler) can be supplied.

Moreover, storing the program in a ROM is simple and cheap operation, particularly if some of the recently available semiconductor integrated ROM's are used [5], [6].

Owing to the reduced interface circuitry and the modular organization of the memories, the computer can be profitably used in different applications, greatly ranging in complexity.

Besides, a versatile organization of input-output facilities allows the requirements occurring in different applications to be easily fulfilled by program, without necessity of large buffers.

As an immediate application the computer has been equipped with a program for two axes, linear and circular interpolation, speed control and tool offset correction [7].

Programs for three-dimensional linear interpolation, two axes interpolation for lathe control and a more sophisticated speed and acceleration control are presently under development.

REFERENCES

[1] - F.W. Wilson, "Numerical Control in Manufactoring". McGraw-Hill, New York, 1963.

[2] - J. Thilliez, "Le Commande Numèrique des Machines". Dunod, Paris, 1967.

[3] - G.B. Gerace, "Digital System Design Automation. A Method for Designing a Digital System as a Sequential Network System". IEEE Trans. on Computers, C-17, November 1968.

[4] - V. Casarosa, "Un Programma per l'Automazione del Progetto Logico di Sistemi Numerici". Presented at Congresso A.I.C.A. 1968, Naples, September 26-29, 1968.

[5] - G.L. Bentivoglio, F. Forlani, N. Minnaja, G. Sacchi, "Matrice Integrata di Diodi Schottky per Memoria Fissa". Presented at XV Rassegna Internazionale Elettronica e Nucleare, Rome, April, 1968.

[6] - D.H. Chung, "A Monolithic High Speed Read-only Memory". IEEE Int. Conv. Digest, 31, 1968.

[7] - P. Maestrini, "Sulla Determinazione e Interpolazione della Traiettoria dell'Utensile nelle Macchine a Controllo Numerico Continuo". To appear in Calcolo.

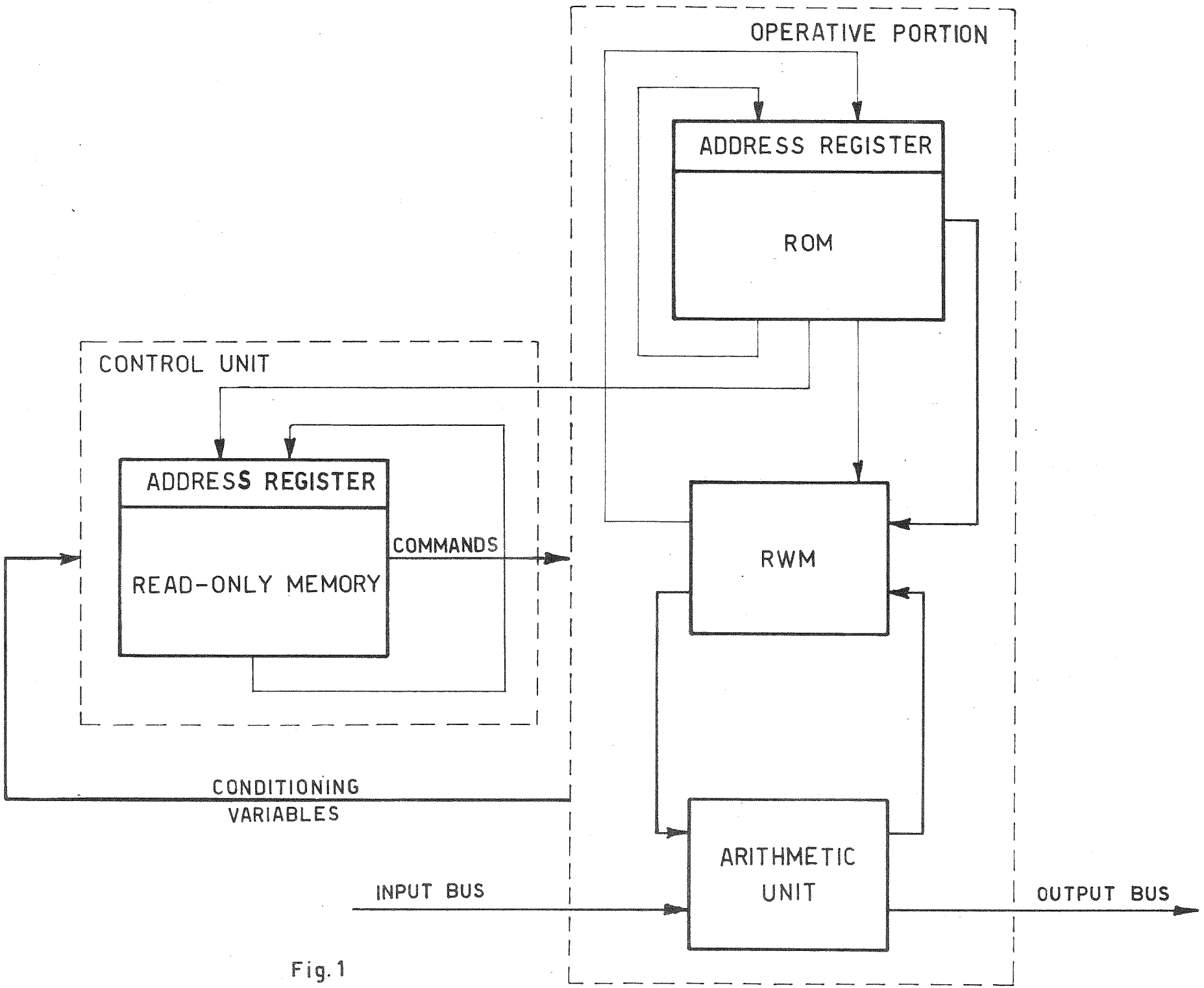
OPERATIVE PORTION
ADDRESS REGISTER
ROM
CONTROL UNIT
ADDRESS REGISTER
READ-ONLY MEMORY
COMMANDS
RWM
CONDITIONING
VARIABLES
ARITHMETIC
UNIT
INPUT BUS
OUTPUT BUS

Fig. 1

MICRO-ORDER FORMAT

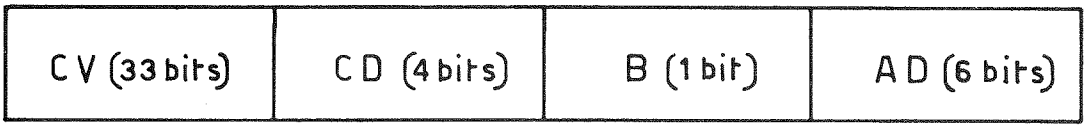

Fig. 2