Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Use of the Group Operations, Fractal Channels and Switching of Consecutive Channels in the Multiprocessing Control Systems

G.G.Stetsjura, M.F.Karavay

Institute of Control Sciences RAS, Russian Federation, Moscow
(e-mail: stetsura@ipu.rssi.ru)

We present here three components (Group Operations, Fractal Channels, formation of the serial channels by comparators) of the new concept – the method "Calculation in the Common Channel." This method allows executing the distributed calculations in the course of data transmission, and accelerates many types of calculations in control systems.

1. INTRODUCTION

The method "Calculation in the Common Channel" (CCC or BOK in Russian) has been developed in Institute of Control Sciences RAS for multiprocessor and multicomputer control systems, and for local area networks (Prangishvili et al., 1984), (Stetsjura,. 2005). We shall use in this context a "computing unit" or abbreviation "CU" for "processor" and "computer". Basic part of CCC is the Group Operations (GOs). The GOs carry out pipeline calculations on a chain of CU with data that are transferring through the bit-serial channel from one CU to another.

The GOs execute distributed calculations in process of the data transfer through the channel without need of additional time as compared with the transfer time. Therefore, the calculation time has a little dependency on a quantity of processors participating in the operation. The GOs also allow reducing essentially energy dissipation in the processors of the systems.

Section 6 of this paper describes the Fractal Channel (FC). It is the bit-serial channel that in the regular way divides system of CU onto subsystems. Each subsystem is connected to the system in the only one point using short lines. In presence of failures the regularity of FC facilitates and speeds up recovery of the system.

In section 7 the formation in dynamics of the bit-serial channels connecting CU for the CCC-method is described. The offered solution essentially accelerates the decision of many applied tasks.

2. GROUP OPERATIONS

Let us begin the GO presentation from the description of processors used for the CCC method, we call it the CCC processors. A message with the command for CCC to perform Group Operation is moving from one CU of the channel to another. Specialized CCC processors are sat in the CUs of the channel and are controlled by the CU. The GO is a two arguments operation. The operation is executed in each CU as follows. One argument x for operation comes from the message, and another argument y for operation comes from the CU. As the command message reaches the CU, the CCC processor synchronously executes the one-bit operation. Its result replaces one-bit operand x in the serial message, and message is moved to the next CU. I.e. the result obtained in one of the CUs becomes the operand x for the CU that follows one, etc. On passing through the entire channel, the message will contain the CCC result that is common for all CUs. The channel is represented by the pair of transmission lines "1" and "0" with directed transfer of the signals. The CCC processor contains a switch and handles data in the channel using this switch, which is placed in the break of the lines of the channel. A binary "1" in the channel is designated by presence of the signal in the line "1" and the binary "0" is designated by presence of the signal in the line "0". The switch can be put into one of the four states shown in Fig.1. The inputs of the lines to the switch are at the left and the outputs are at the right.

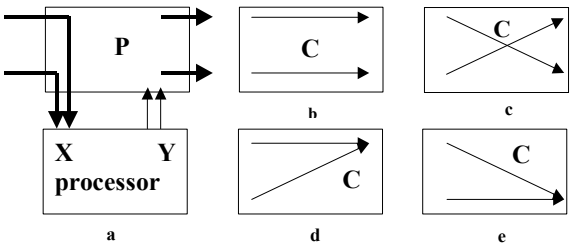

Fig. 1. Group operations

Signals are transmitted through the switch without an additional delay. The switch is transferred to one of the four states (*b, c, d, e)* by the CU. If a binary value of x is delivered from the channel to the input of the switch, which is in the state c, then the switch outputs a negation of x. If the switch is in the states d or e it transforms any values of x to constant values

"1" or to constant value "0", respectively. If the switch is in the state *b* a signal is moved from inputs to outputs as it is.

Let us consider the fulfillment of logical and arithmetic group operations in the CCC processor.

Exclusive OR

The processor turns the switch to the state *b* or *c* if $y = 0$ or $y = 1$, respectively. In this case, the switch is functioning as exclusive OR for the variables *x* and *y*.

This operation and all other GOs have the important property: there is no need to wait the arrival of the operand x from the channel to turn the switch to the required state. It is sufficient to know the value of y. Because the switch can be set up before an arrival of x, the CCC operation is performed without delay. If we disregard the length of the line, then the time it takes to perform operation on one processor or a group of processors is identical.

Let us consider a few other GOs operations.

Logical multiplication $R = x \wedge y$

If $y = 0$, then R = 0 and the switch must be turned to the state *b*. Otherwise, it must be in the state *c*.

Logical addition $R = x \vee y$

If $y = 1$, then R = 1 and the switch must be turned to the state *d*. Otherwise, it must be turned to the state *b*.

Count

The operand is transmitted over the channel, starting with less significant bit. The "exclusive OR" operation is performed. To take into account a carry to the next bit if there is one, and to check if the carry exists, there is time until the arrival of the next bit. So, the count function is also executed without delay.

Maximum of two numbers *X* and *Y*

The operand is transmitted over the channel, starting with the most significant bit. If the higher order bit of *Y* equals to 1, then it is necessary to switch the state *c*, otherwise *b*. If the higher order bit of the number *Y* equals to 0 and the higher order bit of X equals to 1, then the given processor ceases the operation and leaves the switch in the state *b*. This procedure is applied to all bits.It is clear that after the signals of all bits pass through all switches of the CCC processors, the maximum number of the numbers stored in all CCC processors will be found.

The finding of a minimum (the min operation) is performed by simple replacement of the values 0 and 1 in the algorithm for the max operation. The calculations with the CCC, which are carried out without delay, are described in more details in (Stetsjura, 2003, 2005).

3. IMPLEMENATION OF THE CCC ELEMENTS

It is worthy to note about possibility to implement the CCC with the use of only one line (Stetsjura, 2004b). For this, it is necessary to use an optical polarized signal and to perform the negation, rotating its polarization plane.

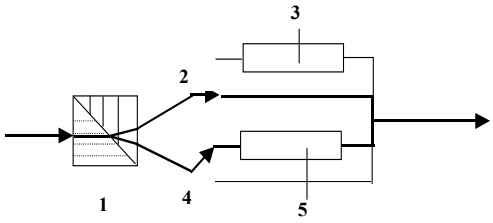

Fig. 2 Optical element

Such implementation of CCC is demonstrated in fig. 2. The device comprises a double-refracting polarizer, Signal leaving a splitter outputs to different paths, depending on the entering signal polarization: to the switch 2, or 4. Thus, the signals with values 1 and 0 are spatially divided and can individually be processed. Let us consider that the entering signal "1" follows to the switch 2 and the signal "0" follows to the switch 4.

The state of the switches shown in fig. 2 leads to transformation of any entering signal to the signal "1". Really, if the device has received a signal "1" the signal comes to the switch 2 and leaves the device without change. The signal "0" comes to the switch 4 and then to element 5. The element 5 changes polarization and inverts value of an entering signal. In this case the value of the output signal of the device is also "1". The switch 2 should be in the top position, and the switch 4 in the bottom position to obtain an output signal equal to "0". For inversion of an entering signal both of the switches should be in the top position.

Note that the described elements show an opportunity to achieve the real zero time delay at carrying out the calculations.

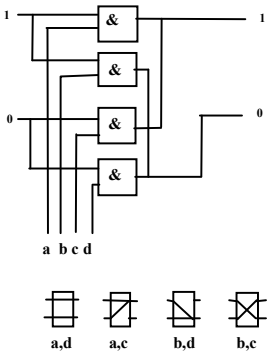

Fig. 3 Electronic element

CCC also can be designed using usual logic elements, as shown in fig. 3. In this design, an operation delay is decreased as well by the following reasons. Firstly, the whole chain of CCC-elements may be considered as delay line, in which many signals can be present simultaneously in different spatial locations. Secondly, the final preparation of CCC-elements for fulfillment the operation is completed before arrival of an input signal from the channel.

4. ENERGY OF THE CCC-ELEMENTS

Let's show distinction in work of traditional logic elements and logic elements CCC. The traditional logic element analyzes the signals and sends the new signal created by this logic element using power source connected to it. The CCC element is turned to the necessary state before appearing an input signal. After that an input signal passes through CCC element, and energy of the signal is not used to switch the elements. The input signal energy is only consumed in the amount necessary for its analysis by the processor. The signal with remaining energy moves to the other CCC-elements or is removed from the system.

There are two important fields of application where energy consumption of the CCC-element functioning is an issue. The first one is multiprocessor integrated systems (system on crystal), which are interconnected by channel and carry out the group operations. In the case the system can consume a powerful signal formed outside of the system, and residual energy of the signal can be removed from the system avoiding the system excessive heating.

The second application field is various systems of environment monitoring. Such systems have sensors remote from the means of data gathering. Usually powerful power supply is required for such sensors only for forming the powerful output signals. But in our case we use only a powerful external signal which also allows the sensors to carry out collective interaction by means of Group Operations (Stetsjura, 2004a).

5. GROUP COMMANDS, DISTRIBUTED PROGRAMS

Group Operations allow to organize execution of the Distributed Programs, (the sequences of Group Commands) (Stetsjura, 2005). The Group Commands (GC) is created using the Group Operations. The structure of the GC is shown in fig. 4.

|operation code | operands |1| result |1| result |0| no result

Fig. 4 Group command

GC has a head (operation code) which defines required CU reactions on GC and addresses or names of the CUs, which are receiving the result of the actions. GC defines a way of operand processing, a location of data and their value, a location of the result. Besides, variants of the operation completion, formation and transfer of the receipt to source and transfer of the master rights to other CUs can be set.

GC size is invariable, if a command has constant quantity of operands. Otherwise, the GC has variable length.

Sometimes it is needed in GC to transfer control to another CU. This CU will be the initiator of another distributed program.

GC defines the chain of operand blocks and blocks of the operation result. The CU, having received from GC an operation code, executes the operation as a sequence of the usual or the Group Operations.

GC is executed by group of CUs. A result of the Group Command execution remains in the CU, or the CU places it into GC. In the last case GC should be executed without time delay. In the case the quantity of CUs has no influence on GC execution time.

A result created by one CU, serves for another CU as an operand, which should be replaced by a new result.

A sequence of GC transmitted into channel is a distributed program.

In case of the control transfer in the program may be more then one processor wants to be the master, and it is needed to choose one of them with the highest priority.

It is a problem of priority access to a channel. The problem can be solved with operation "maximum" in which priorities of CUs are comparing. As a result, a new leader will be chosen.

6. THE FRACTAL CHANNELS ORGANIZATION

Structure of FC is similar to structure of the Serpinsky curve (Crownover, 1995). Fig.5 show the Serpinsky curve (Sk) obtained as a result of the sequence of iterations 0, 1, 2, 3 with the use of the L-system.

The nodes of the Sk are connected by edges that are the curve segments. Each iteration is fulfilled by adding branches to the curve constructed on the preceding iteration. Each branch returns to the point of its insertion. The plotted Sk is a closed one and covers without intersections all the area it occupies. The Sk reveals the specified local feature - it has been filling the area of the successive branch and only after that it passes to fill the next one. The Sk is a topologically equivalent of a ring.

Let's construct the flat FC similarly to the Sk. We locate CU into the FC nodes. They are denoted by heavy spots in Fig. 5a. The number of CU and the nodes in which they are located depends on the requirements of the specific implementation. If there is no CU in the node, then such a node is just a bend in the line that enables us to locate it on the plane in the required manner.

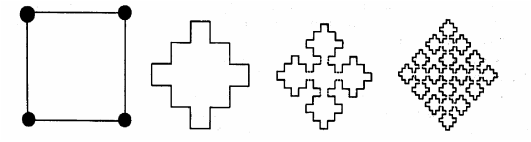

Fig. 5 Fractal channels

Each CU has a switch. The FC nodes are joined by additional communication links, which are shown in Fig.6 as dotted and dash-dot links. The dotted links serve to break up the system into parts operating independently. By closing a dotted link, the switch disconnects the branch joined to this link from the rest of the FC. In this case, the communication system is broken up into some parts, which allow an independent data exchange. If faulty FC parts appear, then the switches can isolate them by disconnecting and creating of a bypass route along dash-dot link. Because the FC represents actually a ring connection it may be constructed as fault tolerant ring as for specified parts so completely for FC as a whole (Karavay, 2005).

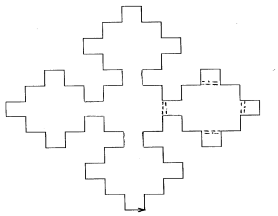

Fig. 6 Additional communication links

Instead of the single line, a group of communication lines can be laid in a similar way to increase the throughput. It is possible to construct FC with different length of edges and various levels of the iteration in different parts of the line. This permits us to construct FC according to requirements of a specific device (Stetsjura, 2005).

7. USE OF THE GROUP COMMANDS

7.1.Checking of the system integrity.

For this checking it is necessary to clarify whether all processors are in proper operating condition, and in case of faults occur determine their locations. To do this, a message containing a control group command is sent over the channel. This command contains the field for result of the serviceability check. Each serviceable processor adds “1” to the content of this field, which is initially equal to zero, using the CCC. The field will display the number of serviceable processors after this operation passes over the entire channel. The same operation includes the fields for addresses, into which each serviceable processor inserts its address (the sequential number in the channel), replacing the preceding content of the field. Each serviceable processor verifies whether this field contains the address of its immediate neighbor. If this address is unavailable, the processor fixes the presence of an error in one or in several its predecessors. After all processors fulfill the control operation, the complete information is available, which is necessary for removing faults or isolating them by bypassing routes using the switches of the channel.

7.2. Conflict elimination at the access to the common resource.

In real time systems the Group Commands allow to quickly eliminate conflicts, which arise from the simultaneous demand of several sources to access the same receiver at the unpredictable moments. For the elimination of these conflicts the GC "elimination of conflict" is sent.

The source, before access to the receiver, checks if the command contains the name of this receiver. If this name is not present, the source inserts the receiver name in the GC, and adds a field-counter for this name. If the command contains the receiver the source increases the counter of this name by “1”, using CCC-addition. After the command passes all sources, each of them will know its serial number in the queue. This queuing excludes conflicts.

7.3 Secure delivery of the broadcasting and group messages.

For confirming that the message is delivered each CU, which has received the message, adds “1” to the result block of the GC , using the CCC-count. The source of the message receives this sum, and determines presence of an error with high speed.

7.4 Queue reconstruction

If some CUs concurrently wish to execute a work, they must organize a queue taking into account the participant priorities. It is worthy to note the advantage of using GC for queuing. If some participants leave a queue there is no need to take special actions for queue rebuilding. Group Commands restore the queue automatically.

7.5 GOs and collective operations of the MPI standard

Typical collective operations of MPI interface are shown in fig.7. The MPI is used for handling parallel calculations. MPI became the international standard. Collective operations of MPI are identical to the group operations, as shown in fig. 7. But MPI operations usually are executed by processor programs, whereas the group commands have special technical support and they are executed many times faster.

Recently the some firms start to manufacture technical means for MPI support, for example, IBM for BlueGene/L. The BlueGene/L executing many of computing operations executes only one operation of MPI. This is an evidence of complexity of the MPI-command implementation. The group commands are more fast. Use of GC for support MPI should greatly decrease load on the systems switches.

Broadcast operations: distribution & collection
reducing operations MPI_SUM – vector sum
MPI_PROD – vector product
MPI_LAND – logical AND
MPI_BAND – binary AND
MPI_LOR – logical OR
MPI_BOR – binary OR
MPI_LXOR – logical exclusive OR
MPI_BXOR – binary exclusive OR
MPI_MINLOC
MPI_MAXLOC

MPI_MAXLOC(U,V,I,J) = (W,K)
W = max (U, V)

$$K = \begin{cases} I & \text{if } U>V \\ \min(I, J) & \text{if } U = V \end{cases}$$

Fig. 7 MPI collective and group operations

8. USE OF THE SWITCHING OF CHANNELS

In a calculation process with GOs the dynamic reorganizations of the CU connections is often required. The switches (commutators) usually carry out this task. We will apply the switches in an unusual mode of operation to create groups of the serial channels with the CU in dynamics. In fig. 8 the channel at the approach to the CU of system passes through the CCC-processor, which executes the GOs. This module works under the CU control. Channel signals pass through the CCC processor without any essential delay. In fig. 8 ovals designates the CCC processors, and a rectangle does the switching. The CCC processors grouped in the channel 1, 5, 3, 4, 2, 6.

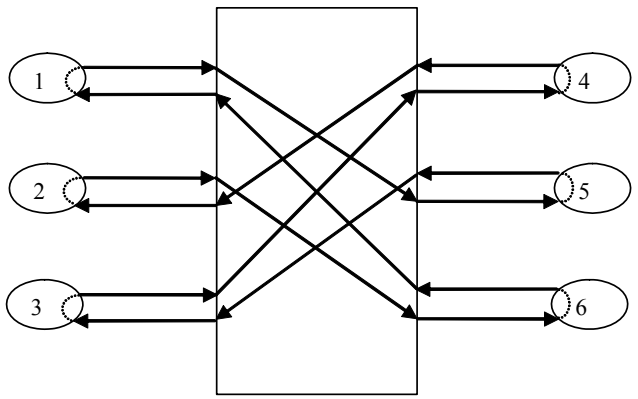

Fig. 8 Serial channel with switch

Instead of the single channel, the switch can create a group of the channels. For example, instead of the channel 1, 5, 3, 4, 2, 6 two channels 1, 5, 3 and 4, 2, 6 may be created.

The system switch must be the nonblocking, one-step switch.

For such systems with switches a number of typical problems of the linear algebra, the signal processing, and sorting have been investigated, and give the acceleration of calculations with factor $\log_2 n$ in comparison with known results (Stetsjura, 2008a).

The described connecting structure with switches has allowed creating the fast distributed algorithms of attainment of mutually coordinated decisions in the hard real-time systems (Stetsjura, 2008b). It is some simplified version of the Byzantine fault tolerance problem. Utilization of a small quantity of GOs is enough for detection and neutralization of errors. In particular, it simplifies very much backoff in calculation.

9. CONCLUSION

This article shows the possibility to achieve two important properties of control systems. 1. Many distributed calculations can be accelerated using of setting the logical elements just before the calculation. 2. We revealed the new possibility to decrease the energy dissipation in the data processing devices: the signals are not generating in the devices, they only change their route. The results presented here have no analogues.

REFERENCES

Crownover R. (1995) *Introduction to Fractals and Chaos*. Jones and Bartlett Publishers, Boston. pp. 352.

Karavay M.F. (2005) Fault-Tolerant Design For Hamiltonian Target Graphs / Proceedings of IEEE EAST-WEST Design&Test International Workshop. Odessa: Sept. 15-19, 2005. P. 175-181.

Prangishvili I.V. Podlazov V.S. Stetsjura G.G. (1984). *The Local Microprocessor Computing Nets* (in Russian). Nauka, Moscow. pp.176.

Stetsjura G.G. (2003). Possibilities of the Application of Fractal Lines and Grouped Operations in Multiprocessor Systems with a Rearrangeable Structure for Evolutionary Computations. *Automation & Remote Control* n.12. p. 164-178.

Stetsjura, G.G.(2004a) A way of high-power output signals formation by micropower sources. *Datchiki I Sistemi* (*Sensors & Systems*, in Russian). n. 12. p. 51.

Stetsjura G.G. (2004b). Potentialities of Physical Realization of Cellular Automata with Remote Connections and Group Operations. *Automation & Remote Control*. n.8. p.174-184.

Stetsjura G.G. (2005). *Methods of Overlapping of Calculations and Data Transmission in Multiprocessing Systems and Local Networks* (in Russian). Institute of Control .Sciences, Moscow pp. 86 http://www.ipu.ru/labs/lab31kom/ggs.zip

Stetsjura G.G. (2008a) Combination of calculations and data transmission in systems with switches. *Automation & Remote Control*. n.6.

Stetsjura G.G. (2008b) The fast distributed algorithms of attainment of mutually co-ordinated decisions in the hard real-time systems. *Automation & Remote Control*. (to be published)