SIMD SYSTEM HAVING LOGIC UNITS
ARRANGED IN STAGES OF TREE
STRUCTURE AND OPERATION OF STAGES
CONTROLLED THROUGH RESPECTIVE
CONTROL REGISTERS

Inventors: Tatsuya Shindo, Kawasaki; Kaoru
Kawamura, Yokohama; Masanobu
Umeda, Manakata, all of Japan;
Toshiyuki Shibuya, LA, Calif.; Hideki
Miwatari, Yokohama, Japan

Assignee: Fujitsu Ltd., Kawasaki, Japan

Filed: Jul. 19, 1995

Related U.S. Patent Documents
Reissue of: Patent No. 5,230,087
Issued: Jul. 20, 1993
Appl. No.: 07/409,613
Filed: Sep. 19, 1989

Foreign Application Priority Data
Sep. 19, 1988 [JP] Japan 63-234545
Sep. 19, 1988 [JP] Japan 63-234546

Int. Cl. 7/12/11; 712/12; 712/16; 712/20

U.S. Cl. 7/12/11; 712/12; 712/16; 712/20

Field of Search
395/200.3, 200.38, 395/200.43, 200.68, 200.82, 800.01, 800.1,
800.12, 800.16, 800.21, 800.22, 825, 840;
364/131-134, 736, 741; 709/200, 205, 213,
238, 252; 712/1, 12, 16, 20, 21, 22, 710/5,
20; 700/2, 3, 4, 5; 708/534, 490

References Cited
U.S. PATENT DOCUMENTS
4,304,002 12/1988 Hunt 708/104
4,541,048 9/1985 Proposition et al. 712/32
4,574,345 3/1986 Konesky 395/559
4,621,339 11/1986 Wagner et al. 712/22

FOREIGN PATENT DOCUMENTS

OTHER PUBLICATIONS
1-8, C. E. McDowell.

"Architecture & Applications of the Connection Machine"

European Search Report conducted at The Hague by Examiners

Primary Examiner—Larry D. Donaghe
Attorney, Agent, or Firm—Staas & Halsey I.I.P

ABSTRACT
In a parallel computer system using a SIMD method
controlled by a controller and a plurality of processor elements,
each of the processor elements has a storage unit to store
data to be processed, the controller controls operation of
the processor elements, and the parallel computer system
performs processing of the data based on a calculation control
signal transmitted from the controller. The parallel computer
system further has a data collection unit connected between
the processor elements and the controller for receiving output
data from the processor elements, performing a predetermined
calculation, and outputting calculated data to the
controller; and a calculation control unit connected between
the data collection unit and the controller for transmitting
the calculation control signal from the controller to the data
collection unit to make it possible to perform the prede-
termined calculation in the data collection circuit.

11 Claims, 25 Drawing Sheets

---

[Diagram of calculation control circuit and data collection circuit]
Fig. 1

CALCULATION CONTROL CIRCUIT

15A 15B 15C 15D

17 SIGNAL LINE FOR CALCULATION

18 SIGNAL LINE FOR PROCESSOR

14

DATA COLLECTION CIRCUIT

15A 15B 15C 15D

17 SIGNAL LINE FOR CALCULATION

18 SIGNAL LINE FOR PROCESSOR

10

CONTROLLER

CONTROL MEMORY

GLOBAL DATA REGISTER

11

12

13

14
Fig. 3

GLOBAL DATA INPUT

GT 30

ALU

CT 31

DATA REGISTER

PE 14

N PORT

W PORT

E PORT

CONTROL SIGNAL

INPUT

COLLECTION DATA OUTPUT

S PORT
Fig. 4

OR CALCULATION CIRCUIT

AND CALCULATION CIRCUIT

MAX/MIN/ADD CALCULATION CIRCUIT

SEL

GLU

XGCR

GNOPS

GLSTS

GMAXS
<table>
<thead>
<tr>
<th>SIGNAL NAME</th>
<th>CONTENTS</th>
<th>ATTRIBUTE</th>
<th>bit</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLI</td>
<td>GLU INPUT DATA</td>
<td>I</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>GLO</td>
<td>GLU OUTPUT DATA</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>XGOPS</td>
<td>GLU OPERATION CODE</td>
<td>I</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XGCR</td>
<td>CARRY CLEAR (0: CARRY CLEAR, 1: NORMAL)</td>
<td>I</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GLST S</td>
<td>GLU INPUT STAGE SW (00:32, 01:16, 10:8, 11:4)</td>
<td>I</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>GMAXS</td>
<td>MIN, MAX SELECTION SIGNAL (0: MAX, 1: MIN)</td>
<td>I</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GNOPS</td>
<td>&quot;O:&quot; FORCELY INPUT SIGNAL (0:&quot;O&quot; INPUT, 1:&quot;NORMAL&quot;)</td>
<td>I</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td></td>
<td></td>
<td></td>
<td>40 bits</td>
</tr>
<tr>
<td>CONTROL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fig. 7

```
<table>
<thead>
<tr>
<th>OP1</th>
<th>ADD CARRY CLEAR</th>
<th>ADD NORMAL</th>
<th>ADD NORMAL</th>
<th>ADD NORMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1 (INPUT)</td>
<td>BO (LSB)</td>
<td>B1</td>
<td>B2</td>
<td>B3 (MSB)</td>
</tr>
<tr>
<td>OP2</td>
<td>ADD CARRY CLEAR</td>
<td>ADD NORMAL</td>
<td>ADD NORMAL</td>
<td>ADD NORMAL</td>
</tr>
<tr>
<td>D2</td>
<td>BO</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
</tr>
<tr>
<td>D3 (OUTPUT)</td>
<td>BO</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
</tr>
</tbody>
</table>
```

1 CLK
Fig. 10

[Diagram of a control system with interconnected blocks labeled SC, RAGC, and PE, along with control groups identified as G1, G2, G3, and G4.]
**Fig. 12**

FIRST MEMORY SPACE (OBJECT AREA) 300

SECOND MEMORY SPACE 300

REAL PE GROUP 301

**Fig. 13**

n WINDOWS

m WINDOWS

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>…</th>
<th>n-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>n+1</td>
<td>n+2</td>
<td>2n</td>
<td>2n-1</td>
</tr>
<tr>
<td>2n</td>
<td>2n+1</td>
<td>2n+2</td>
<td>3n</td>
<td>3n-1</td>
</tr>
</tbody>
</table>

WINDOW 302

SECOND MEMORY SPACE 300
**Fig. 14A**

```
MEMORY SPACE OF VIRTUAL PE
000
FFF
000
MEMORY SPACE OF VIRTUAL PE
FFF
000
MEMORY SPACE OF VIRTUAL PE
```

**Fig. 14 B**

WINDOW NUMBER

```
aaaa 0000 (BASE ADDRESS)
```

+ 

VIRTUAL PE ADDRESS (RELATIVE ADDRESS)

```
0000 bbbbbbbbbbb (16 BITS)
```

REAL ADDRESS

```
aaaa bbbbbbbbbbb
```
Fig. 15

CONTROL GROUPS
G1

32

128

14

32

128
Fig. 16

VIRTUAL PE ADDRESS FROM CONTROLLER

110
SC
G1 32x32

120
G2 32x32

110
G3 32x32

120
G4 32x32

G5 32x32

G6 32x32

G7 32x32

G8 32x32

G9 32x32

GG0 32x32

GG1 32x32

GG2 32x32

G13 32x32

G14 32x32

G15 32x32

G16 32x32
Fig. 21

EVCLR (CLEAR/LOAD)

E

W

N

S
### Fig. 22

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTRATION OUTPUT</td>
<td>CONSECUTIVENESS OUTPUT</td>
</tr>
<tr>
<td>EVENT OUTPUT</td>
<td>SELF - EVENT</td>
</tr>
<tr>
<td></td>
<td>REGISTRATION</td>
</tr>
<tr>
<td>ADDRESS HOLDING</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Note: The diagram shows logical states represented by 'T' for true and 'F' for false.
Fig. 24
**Fig. 25A**

(W \(A-1\))

\(N\)

\(W\)

\(S\)

\((A+B)\)

\(E\)

\((A+1)\)

\((A-B)\)

**CONTROL GROUP**

**Fig. 25B**

**BOUNDARY MARK**

**Fig. 25C**

**PASS**

\(A+B\)

\(A-B\)

\(A+1\)

\(A-1\)

**ADD-REG**

\(B\)

**FIFO OUTPUT**

\(8\)

\(8\)

\(8\)

\(8\)

\(8\)

\(S80\)

\(R80\)

\(R81\)

\(A\)

\(ALU\)

\(B\)

\(S81\)

\(R82\)

\(S82\)

**UPPER ADDRESS BITS**

**WINDOW NUMBER**
SIMD SYSTEM HAVING LOGIC UNITS
ARRANGED IN STAGES OF TREE
STRUCTURE AND OPERATION OF STAGES
CONTROLLED THROUGH RESPECTIVE
CONTROL REGISTERS

BACKGROUND OF THE INVENTION

1. Field of the Invention
The present invention relates to a parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements connected to each other in a lattice configuration.

2. Description of the Related Art
Parallel computer systems are widely used, particularly, in the field of CAD (Computer Aided Design) which necessitates high speed calculation for a LSI (large scale integrated) circuit design. Accordingly, it is desirable to improve techniques to make these processor elements operate more efficiently in order to meet the requirements of high density and high speed LSI.

There are two types of parallel computers based on the connection configuration between the processor elements and the controller. One method is called an MIMD (multiple instruction stream multiple data stream) method which is constituted by a plurality of processor elements and controllers. In this method, each of the processor elements is connected to a corresponding controller, respectively. Accordingly, it is necessary to provide the same number of controllers as there are processors. However, it is difficult to constitute a large scale parallel computer system using this method because a large number of controllers are necessary in accordance with the number of processors, which can be from several tens to several hundreds of processors.

The other method is called an SIMD (single instruction stream multiple data stream) method which is constituted by a plurality of processor elements and one controller. In this method, the controller is connected in parallel to all processor elements. Accordingly, it is possible to constitute a large scale parallel computer which has a large number of processor elements, for example, several tens of thousands of processors. For example, a "Connection Machine" made by Thinking Machines Corporation uses the SIMD method. This system is constituted by several tens of thousands of processor elements.

There are several problems associated with the SIMD type of parallel computer.

A first problem occurs in the synchronization of all the processor elements. In general, two countermeasures are taken for solving this problem. In the first countermeasure, data for obtaining synchronization is exchanged between processor elements through a transmission line. However, it is necessary for all the processor elements to be connected to effectively apply this method. In the second countermeasure a particular signal for obtaining synchronization is output from each processor element. Then, "wired-OR" logic is performed on all of the synchronization signals and the resultant data of the wired-OR is returned to all of the processor elements. However, the number of processor elements is limited in the wired-OR logic approach because a large delay occurs in the wired-OR logic operation.

A second problem occurs in the order of priority use of the bus line. When the number of processor elements reaches from several thousand to several tens of thousands, it is necessary to determined the priority order of use of the bus line.

A third problem occurs in the extraction of essential data from essential processor elements. The essential data is, for example, maximum data or minimum data.

One type of parallel computer system according to the present invention is provided for the solution of the above problems.

The other type of the parallel computer system according to the present invention can control all processor elements so as to effectively and uniformly distribute the processor elements as a load.

SUMMARY OF THE INVENTION

The object of the present invention is to provide a parallel computer system using a SIMD method and enabling high efficiency data processing and high load distribution capability.

In accordance with the present invention, there is provided a parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements, each of the processor elements having a storage unit to store data to be processed, the controller controls the operation of the processor elements, and the parallel computer system processes data based on a calculation control signal transmitted from the controller, the parallel computer system comprising: a data collection unit connected between the processor elements and the controller for receiving data output from the processor elements, performing a predetermined calculation, and outputting calculated data to the controller; and a calculation control unit connected between the data collection unit and the controller for transmitting the calculation control signal from the controller to the data calculation unit to make it possible to perform the predetermined calculation in the data collection circuit.

Further, in accordance with the present invention, there is provided a parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements, each of the processor elements having a storage unit to store data to be processed, the controller controls operation of the processor elements, and the parallel computer system performs processing of data based on a calculation control signal transmitted from the controller, the parallel computer system comprising: a plurality of control groups, each control group being constituted by a number of processor elements divided from a plurality of processor elements, to be utilized as an address control unit; a plurality of scheduling circuits, with a scheduling circuit being provided for each control group and operatively connected to the controller, for receiving and managing an event signal designating an address signal for data to be processed and transmitted from an adjacent control group; and a plurality of real address generation circuits with a real address generation circuit provided for each control group and operatively connected to the controller, the scheduling circuit and the control group, for generating an address signal for data to be processed by a processor element belonging to the control group based on a base address determined by the event signal to be managed by the scheduling circuit and an address signal applied from the controller.
BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings:

FIG. 1 is a basic block diagram of one type of parallel computer system according to the present invention;

FIG. 2 is a block diagram of one embodiment of a parallel computer system shown in FIG. 1;

FIG. 3 is a schematic block diagram of a processor element shown in FIGS. 1 and 2;

FIG. 4 is a schematic block diagram of a gathering logic unit GLU shown in FIG. 1;

FIG. 5 is a table for explaining various control signals shown in FIG. 4;

FIG. 6 shows one example of a data collection circuit having gathering logic units shown in FIG. 1;

FIG. 7 is a signal timing chart for explaining the operation of the circuit shown in FIG. 6;

FIG. 8 is a detailed block diagram of a gathering logic unit shown in FIG. 4;

FIG. 9 is a detailed block diagram of the MAX/MIN/ADD calculation circuit shown in FIG. 4;

FIG. 10 is a block diagram of the other type of parallel computer system according to the present invention;

FIG. 11 is a schematic block diagram of a processor element shown in FIG. 10;

FIG. 12 is a view for explaining the concept of the present invention shown in FIG. 10;

FIG. 13 is a view for explaining the division of the virtual area shown in FIG. 12;

FIGS. 14A and 14B are views for explaining addresses of memory spaces shown in FIG. 12;

FIG. 15 is a view for explaining control groups shown in FIG. 10;

FIG. 16 is a block diagram of control groups and peripheral circuits;

FIG. 17 is a block diagram of for explaining the pseudo processor element;

FIG. 18 is a detailed block diagram of the scheduling circuit shown in FIG. 10;

FIG. 19 is a detailed block diagram of the input circuit for the window number shown in FIG. 18;

FIG. 20 is a detailed block diagram of the consecutiveness detection circuit shown in FIG. 18;

FIG. 21 is a detailed block diagram of the event input circuit shown in FIG. 18;

FIG. 22 is a logic table in the event interpretation circuit shown in FIG. 18;

FIG. 23 is a detailed block diagram of the FIFO circuit shown in FIG. 18;

FIG. 24 is a detailed block diagram of the registration flag circuit shown in FIG. 18;

FIGS. 25A to 25C are detailed block diagram of the address circuit shown in FIG. 18; and

FIG. 26 is a detailed block diagram of the real address generation circuit shown in FIG. 18.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1 is a basic block diagram of one type of parallel computer system according to the present invention. In FIG. 1, reference number 10 denotes a controller, 11 a control memory for storing a micro-code including output control signals, and 12 a global data register for performing an input/output operation of the data processed or to be processed. The control memory 11 and the global data register 12 are provided in the controller 10. Reference number 13 denotes a data collection circuit for collecting output data from processor elements (PE) 14. 15A to 15D denote control registers CR constituting a calculation control circuit and connected to each other using a pipe-line method for applying various calculation control signals to the collection circuit 13. 16A to 16D denote gathering logic units (GLU) constituting the data collection circuit 13 and each constituted by a tree configuration. Reference number 17 denotes a signal line for the calculation control signal to the GLU 18, a signal line for controlling processor elements, and 19 a data line for broadcasting global data.

Each of the processor elements comprises a data register for storing the data to be processed and an arithmetic logic unit (ALU) as shown in FIG. 3. The arithmetic logic unit ALU calculates the data stored in the register in response to the order transmitted from the controller 10 through the signal line 18.

Each gathering logic unit GLU 16A to 16D collects the output data transmitted from the processor elements. The gathering logic units 16A to 16D are connected in the form of a tree configuration having several stages. That is, in FIG. 1, the units 16A are the first stage, the units 16B are the second stage, and the unit 16D is the final stage. The outputs of the processor elements 14 are input to the gathering logic units 16A. The resultant calculation data obtained in the GLU's 16A are output to the GLU's 16B. Similarly, the resultant data obtained in the GLU's 16B are output to the next stage. The final stage 16D gathers all resultant data obtained in the previous stages and the data calculated in the final stage 16D is output to the global data register 12 in the controller 10.

Each of the calculation control registers CR 15A to 15D are connected in series with each other using the pipe-line method. The number of registers is equal to the number of stages in the gathering logic unit GLU. In this case, the calculation in each stage is performed in response to the calculation control signals, for example, an “ADD” calculation signal, transmitted through the signal line 17. That is, when the calculation signal “ADD” is input to the first stage 16A, the calculation suggested by the calculation signal is performed in the first stage 16A regarding the data output from the processor elements. This calculation signal is transmitted to the next stage in response to the clock signal from the controller 10 and the same calculation suggested by the calculation signal is performed in the second stage 16B. The above calculation is performed using the pipe-line method. That is, when the first calculation signal “ADD” is input to the second stage, the next calculation signal, for example, “MAX” is input to the first stage.

The synchronization of all processor elements is performed in accordance with a synchronization signal from the controller 10. The controller 10 sends the synchronization signal to all processor elements through the control line 18 to output the value “1” when each processor element completes the predetermined processing. At the same time, the signal “AND” is transmitted to the control register 15A through the control line 17.

When the calculation signal “AND” is set in the register 15A, the GLU 16A of the first stage performs an “AND” calculation regarding all outputs from the processor elements in response to the first clock. The same “AND” calculation is performed in the GLU 16B of the next stage.
in response to the next clock. When the same “AND” calculation is performed in the GLU 16D of the final stage in response to the clock and the resultant data is the value “1”, the controller 10 can recognize that all processor elements output the value “1”.

The essential processor element having the essential data is extracted as follows. A proper processor number is previously attached to each processor element. First, the controller 10 commands the essential processor element to output the proper number. Second, the controller 10 commands another processor element to output a suitable signal, for example, the value “11-1” or “00-0”. The controller 10 then sends the control signal “MAX” or “MIN” to the control register 15A. Accordingly, the essential processor element can be selected in response to “MAX” or “MIN” of the number in the collection circuit 13. In this case, a next essential processor element can be selected from the remaining processor elements excluding the first essential processor element in the same manner as the above. Accordingly, it is possible to use this circuit to select the priority order of use of a bus line.

FIG. 2 is an embodiment of a parallel computer system shown in FIG. 1. The same reference numbers as used previously indicate the same components. In FIG. 2, reference number 20 denotes a processor array the elements of which are connected to each other in a lattice configuration. As explained above, the processor array and the collection circuit are controlled by the control signals from the controller 10. The controller memory 11 in the controller 10 comprises a plurality of control formats 1 to n.

The controller 10 further comprises a sequencer 21 which determines the sequence for reading out the control information from the control memory 11. The global data register 12 is a register for holding the data transmitted in common to all processor elements and to receive the output data from the collection circuit 13.

FIG. 3 is a schematic block of a processor element. In FIG. 3, reference number 30 denotes a data register for holding the data to be processed. Reference number denotes an arithmetic logic unit ALU for performing calculation using the data stored in the data register 30. The processor element 14 is controller by the same control signal transmitted from the controller 10. This control signal includes an address of the data register 30 and an operation code for the arithmetic logic unit 31. The processor element 14 further comprises four ports, i.e., east port (E), west port (W), north port (N) and south port (S) for communicating between adjacent processor elements. The processor element 14 further comprises an input terminal GT for inputting the data from the global data register 12, and a collection terminal CT for outputting the data.

The processor element 14 is a one-bit type and the input/output operation to the data register 30 is basically performed for each bit. Data larger than one bit is processed from the most significant bit (MSB) to the least significant bit (LSB) for each bit.

FIG. 4 is a schematic block diagram of a gathering logic unit GLU. Each of the gathering logic units GLU 16A to 16D comprises an OR calculation circuit 40, an AND calculation circuit 41, a MAX/MIN/ADD calculation circuit 42 and a selector circuit SEL 43. The signal GLI is data input to the GLU 16 and constituted by 32 bits. This means that one GLU 16 can handle a maximum of 32 bits in the input side. The signal GLO is resultant data and is constituted by one bit.

Kinds of calculation control signals input to the GLU 16 from the control registers 15A to 15D are as follows. These control signals are shown in a table in FIG. 5.
The final stage GLU 16-5 performs the addition calculation for four input signals in response to the order OP-2. The resultant data is output to the register S8. Since the collection circuit is constituted by two stages, the GLU 16-5 performs the same operation as that of the GLU 16-1 to 16-4 after a delay of one clock.

As explained above, the calculation is performed using the pipe-line method. Regarding the calculation of “MAX” and “MIN,” the same calculation as mentioned above can be performed with the optional bit length. In this case, the bits are input from the MSB to the LSB.

FIG. 8 is a detailed block diagram of a gathering logic unit shown in FIG. 4. In FIG. 8, A1 denotes an AND circuit, N1 to N5 denote NOR circuits, NA-1 to NA-5 denote NAND circuits, S1 to S5 denote selectors, and MA00 to MA40 denote MAX/MIN/ADD calculation circuits. The AND circuit A1 sets the input data GLI to all zeroes when the control signal GNOPS is the value “0.” The NOR circuits N1 to N4 and the NAND circuit perform the OR calculation in accordance with the number of input stages regarding the input data GLI. The NAND circuits NA2 to NA5 and the NOR circuit N5 perform the AND calculation regarding the input data GLI. The MAX/MIN/ADD calculation circuits MA00 to MA40 perform the maximum/minimum/addition calculation. The selectors S1 to S4 select the output in accordance with the operation code XGOPS. The selector S5 performs the selection in accordance with the number of the input stage.

FIG. 9 is a detailed block diagram of the MAX/MIN/ADD calculation circuit. In FIG. 9, A10 to A16 denote AND circuits, N10 to N12 NOR circuits, N10 to N16 NOT circuits, O1 to O2 OR circuits, R0 to R1 registers, S10 to S11 selectors, and R0 a full adder having three inputs.

This circuit performs a maximum/minimum calculation for 2 inputs. DM0 and DM1 are the input signals each having one bit to obtain a maximum/minimum value. DA0 and DA1 are the input signals each having one bit for addition. XM is resultant data for a maximum/minimum calculation and XA is resultant data for addition. In the first stage MA00 to MA15 illustrated in FIG. 8, the DM0 is equal to the DA0, and the DM1 is equal to the DA1, respectively.

The operation of the addition “ADD” is explained in detail below. The addition data is input to the DA0 and DA1 bit by bit beginning with the LSB bit. In the first bit, since the signal XGCR is the value “0,” the carry is cleared so that the carry CARRY-0 becomes “0.” The adder 90 performs the addition regarding the DA0 and DA1, and the resultant data XA is output therefrom. When no carry is performed in the addition, the output signal CARRYOUT becomes “0.” When a carry is performed, the output signal CARRYOUT becomes “1.” The signal CARRYOUT is held in the register R0 for use in the addition at the next clock through the selector S10. In the next bit from the LSB, the content of the register R0 is used as the carry CARRY-0, and is added to the DA0 and DA1.

The operation for obtaining the maximum value is explained in detail below. When obtaining the maximum value, the signal GMAXS is the value “0,” the input data is input to the DM0 and DM1 at every bit beginning with the MSB bit. In the first bit, the signal XGCR is “0,” and the outputs of the AND circuit A10 and A11 are “0” so that the output of the NOR circuit N12 becomes “1” and the outputs of the AND circuits A14, A15 and the OR circuit O1 becomes “0.” Accordingly, the selection signal in the selector S11 becomes “10” so that, as the maximum output XM, the output of the OR circuit O2 is selected in the OR logic operation between the DM0 and DM1. In the registers R0 and R1, when one of the DM0 and DM1 previously becomes “1,” the value “1” is set to the corresponding side. That is, when the DM0 is “1” and the DM1 is “0,” the value “1” is set to the register R0. On the contrary, when the DM0 is “0” and the DM1 is “1,” the value “1” is set to the register R1.

When the value “1” is input to one of the registers R0 and R1, the output of the NOR circuit N12 becomes “0” from the next clock. Further, the output of the OR circuit O1 becomes “1” when the register R0 is “1” and becomes “0” when the register R1 is “1.” Accordingly, after the above selection, the selector S11 outputs the value of either DM0 or DM1 in which the value “1” was previously detected.

The operation for obtaining the minimum value is explained in detail below. When obtaining the minimum value, the signal GMAXS is the value “1.” The operation is the same as that for the maximum. When the output of the NOR circuit N12 is the value “1.” The operation is the same as that for the maximum. When the output of the NOR circuit N12 is the value “1,” the selection signal to the selector S11 is “1” and the output of the AND circuit A16 is selected. When one of the DM0 and DM1 becomes “1,” one of the registers R0 and R1 becomes “1” and the selection signal to the selector S11 becomes “00” or “01.” After the above selection, the minimum value in either DM0 or DM1 is selected.

FIG. 10 is a basic block diagram of the other type of parallel computer system according to the present invention. In FIG. 10, reference number 110 denotes a scheduling circuit SC, 120 a real address generation circuit RAGC, and 150 a pseudo processor element. Further, G1 to G4 denote control groups to be used as a control unit for accessing the address. Accordingly processor elements are divided into several control groups. The scheduling circuit 110 and the real address generation circuit 120 are provided for each control group.

The scheduling circuit 110 is a circuit for receiving an event signal to designate the address and for managing the address designated by the event signal by using a queue. The real address generation circuit 120 is a circuit for generating a real address of the data to be processed by the processor element belonging to that control group. This address generation is performed based on a base address determined by the event signal and an address signal applied from the controller 10.

The pseudo processor element 150 is provided in the boundary portion of each control group. The pseudo processor element 150 has a function of sending the data corresponding to the address of the processor element when the processor element located to the boundary portion gives and takes the data between the adjacent processor elements belonging to the adjacent control group. This circuit is provided to ensure consecutiveness between the processor elements.

FIG. 11 is a schematic block diagram of a processor element shown in FIG. 10. This drawing is the same as FIG. 3 except that an external memory 200 is added between the data register 30 and the real address generation circuit 120. The address of the external memory 200 is applied from the real address generation circuit 120 provided in every control group. This type of parallel computer system according to the present invention mainly relates to the address control for the external memory 200.

FIG. 12 is a view for explaining the concept of the present invention shown in FIG. 10. Reference number 301 denotes an actual processor element group, 302 a first memory space.
corresponding to the actual processor element group 301, and 300 a second memory space (virtual area) corresponding to a virtual processor element group. Accordingly, the first memory space 302 coincides with an object area to be processed by the actual processor element group 301. In general, the object area to be processed (for example, a wire pattern area) coincides with the size of the actual processor element group. However, in the present invention, the object area can be widened up to the second memory space. In this case, the actual processor element group 301 moves to the second memory space 300 so that it is possible to process data regarding the larger object area exceeding the first memory space. Therefore, although the virtual processor element group does not actually exist it is possible to obtain the same performance as the processor element group having the second memory space 300 by moving the actual processor element group 301.

FIG. 13 is a view for explaining division of the virtual area shown in FIG. 12. The second memory space 300 is divided into a plurality of windows (mn window). Accordingly, one window corresponds to the first memory space 302 processed by the actual processor element group 301. The window number is attached to each window from 0 to n-1, respectively.

FIGS. 14A and 14B are views for explaining addresses of memory spaces. In FIG. 14A, the external memory 200 of one processor element 14 is divided into sixteen memory spaces for the virtual processor element. That is, “0000” to “FFFF” are addresses for the external memory each having sixteen bits. While, “000” to “FFF” are addresses for the virtual area each having twelve bits. Accordingly, one actual processor element functions as sixteen virtual processor elements.

In FIG. 14B, the window number denotes the base address indicating the head of each memory space of the virtual PE (processor element) and constituted by eight bits “aaaa0000” since a maximum of 256 windows can be provided. Since the external memory 200 is divided into sixteen blocks in this embodiment, the lower four bits are set to “0000”. The virtual PE address “0000 bbbbbbbbb” denotes the relative address of each memory space of the virtual PE. The virtual PE address is transmitted in common to all processor elements from the controller 10. The virtual PE address has “0000” in the upper bits in accordance with the number of the window. As explained in FIG. 14A, when the number of the window is sixteen, the virtual PE address is constituted by twelve bits. As shown in FIG. 14B, the real address “aaaa bbbbbbbbb” having sixteen bits of the external memory 200 can be obtained by adding (or performing an OR operation) the base address and the virtual PE address.

The processing of the data in the virtual PE is performed in such a way that the real PE sequentially processes the corresponding data in the virtual memory space divided from the real external memory 200. In this case, as the simplest method, there is a method in which the real PE always sequentially processes all virtual PE’s including its own external memory. However, this method is not efficient because the virtual PE’s in which the processing is not necessary are included. Accordingly, the method of the present invention selects the virtual PE’s in which the processing is necessary so that the efficiency of the processing can be raised. Therefore, the concept of the “event” is employed to realize this method in the present invention.

The event is started when the conditions to be processed to the virtual PE are realized. The virtual PE which received the event is handled as the object to be processed by the real PE. The controller determines the content of the event in accordance with a program.

FIG. 15 is a view for explaining control groups shown in FIG. 10. As shown in the drawing, the processor elements (PE) 14 are divided into the control groups G1, G2. For example, the PE’s of 128x128 are divided into sixteen control groups G1 to G16 each having 32x32 PE’s.

FIG. 16 is a block diagram of control groups and peripheral circuits. In FIG. 14, G1 to G16 are control groups, (SC) is the scheduling circuit provided for each control group, and 120 is a real address generation circuit also provided for each control group. The scheduling circuit 110 receives the event from the PE and manages the virtual PE to be processed. The virtual PE number to be processed, i.e., the window number, is queued in the scheduling circuit 110 and sequentially processed from the head of the queue. The scheduling circuit 110 sends the base address corresponding to the virtual PE to the real address generation circuit 120. Accordingly, the scheduling circuit 110 performs the queuing and assigns the real PE.

The real address generation circuit 120 generates the real address based on the relative address of the virtual PE and the base address. In this case, the relative address indicates a kind of control signal transmitted in common from the controller to all PE’s, and the base address is determined by the scheduling circuit 110. The real address is transmitted to the real PE’s in each control group.

The scheduling circuit 110 is connected to four adjacent scheduling circuits. Each input/output signal is explained below.

Event signal (as input signal)

This event signal is obtained by the OR logic among the event signals transmitted from all PE’s (32 PE’s in this embodiment) located on the boundary of the control group, and is used as the input signal. This signal is one bit for four directions of E, W, N, and S.

Window number signal (as input signal)

The window number signal of the adjacent scheduling circuit 110 is input as the input signal. The window number signal has eight bits as shown in FIG. 14B for four directions of E, W, N, and S. The scheduling circuit inputs the corresponding window number to the event signal when that event signal is activated, and performs the queuing.

Self-event signal (as input signal)

This signal is obtained by the OR logic among all event signals of the PE’s included in its own control group, and has one bit.

Window number signal (as output signal)

This signal is the window number signal output to the adjacent scheduling circuit 110, and has eight bits for four directions of E, W, N, and S.

Base address signal (as output signal)

This signal is an output signal to the real address generation circuit 120 indicating the corresponding address to the window number of the virtual PE readout from the head of the queue.

Various control signals (as input/output signal)

These signals are output or input signals to or from the controller 10. For example, the control signal NEXT is a
signal to indicator reading out a next virtual PE from the queue, and the control signal DIR is a signal to indicate the direction of the data flow in four directions E, W, N, and S. The control signal EMPTY is a signal to indicate vacancy of the input signal, the clock signal, and the queue.

FIG. 17 is a block diagram for explaining the pseudo processor element (PE) shown in FIG. 10. In FIG. 17, the boundary BD of the control group is provided between the processor PE 14A and 14B. That is, the PE 14A is adjacent to the PE 14B. The pseudo PE (PS-PE) 150A is provided adjacent to the PE 14A, and the pseudo PE 150B is provided adjacent to the PE 14B, respectively.

The pseudo PE is provided for ensuring the consecutiveness of the processing between adjacent control groups. This is because the adjacent control group cannot receive the necessary value of the window when the object window between the adjacent control groups is different. Accordingly, as shown in FIGS. 10 and 17, the pseudo PE is provided to each end of the row of the PE’s in each control group. Therefore, when the object window is consecutive between adjacent control groups, the pseudo PE’s are not used and the PE 14A directly accesses the PE 14B by switching the selectors S1 and S2.

When the PE 14A performs the read/write (R/W) operation to its own external memory 200A, the write data is simultaneously written to the external memory 200a belonging to the pseudo PE 150A. When the PE 14A transmits the data to the PE 14B, the pseudo PE 150A reads the data from the external memory 200a and transmits that data to the PE 14B through the selector S1 instead of the PE 14A.

The address of the external memory 200a is the window address of the PE 14B side. The same operation as the above is performed in case of the data transmission from the PE 14B to the PE 14A. Although this drawing shows the connection of one direction as the lattice of one dimension, it is possible to connect two directions as a lattice of two dimensions.

FIG. 18 is a detailed block diagram of the scheduling circuit shown in FIG. 10 and FIGS. 19 to 25 are detailed circuits of the diagram shown in FIG. 18. In FIG. 18, reference number 500 denotes an input circuit for the window number, 510 a registration table, 520 a consecutiveness detection circuit, 530 an input circuit for the event, 540 an event interpretation circuit, 530 a first-in/first-out (FIFO) circuit, 560 a registration flag circuit, 570 an address holding circuit, and 580 an address calculation circuit. Further, R1 to R4 denote registers for the pipe-line control.

The input circuit 500 inputs the window number determined from the four adjacent directions E, W, N, and S, where DIR is the control signal for indicating the data flow. This circuit is shown in detail in FIG. 19.

In FIG. 19, R10 denotes a register for holding the window numbers input from four directions E, W, N, and S. S10 denotes a selector for selecting the window number in response to the control signal DIR and outputting the selected window number having eight bits.

The registration table 510 is a table for storing flags indicating whether or not the window number is registered. One bit is assigned to each window in a maximum of 256 windows. Accordingly, the window number from the input circuit 500 becomes the address in the table 510. Therefore, double registration of a window number is prevented by this method.

The consecutiveness detection circuit 520 determines the consecutiveness between the present area and the adjacent area. The detailed circuit is shown in FIG. 20.

In FIG. 20, COMB denotes a comparator, 600 an encoder (ECD), OR an OR circuit, and S20 a selector (SEL). CE, CW, CN and CS denote registers for storing the resultant data of the detection of the consecutiveness until the reset signal is input. The comparator COMB compares the upper bits of the address of its own control group with the window number input from the input circuit 500. When the former coincides with the latter, the encoder 600 outputs an enable signal in response to the direction control signal DIR. The enable signal is stored in the registers CE, CW, CN and CS as consecutive data, and the consecutiveness data C-FLAG is output from the selector S20 in response to the control signal DIR.

The event input circuit 530 receives the event signals from four directions. The detailed circuit is shown in FIG. 21.

In FIG. 21, EVCLR denotes an event clear signal to clear each register. S30 denotes a selector circuit. The register R is cleared by the event clear signal EVCLR. When the event signal is loaded in the register R, the event signal is output from the selector S30 through the AND circuit.

The event interpretation circuit 540 judges whether or not the queuing of the window number should be performed, or whether or not the present address should be held. The detailed logic table for determining the output from this circuit 540 is shown in FIG. 22.

In FIG. 22, T denotes an active state of the signals. The registration signal REG indicating the queuing of the window number is output from the circuit 540 only when the output of the input circuit 530 is active. The address holding signal AHS is output when the consecutiveness signal C-FLAG and the event signal are active. Further, the address holding circuit is output when the self-event signal is active.

The FIFO 550 stores the window number to be processed in accordance with the event signal. The detailed circuit is shown in FIG. 23.

In FIG. 23, MEM denotes a memory having the capacity of 8x256 bits, R40 to R43 registers, S40 a selector, WCNT a write counter to output the write address, RCNT a read counter to output the read address, and COMB a comparator. When the registration signal is set in the register R41, the window number stored in the register R40 is written to the address indicated by the write counter WCNT in the memory MEM. Further, the content of the address of the memory MEM is read out in response to the control signal NEXT through the register R42 and the AND circuit, and output through the register R43. When the comparator detects coincidence between the content of the write counter WCNT and the content of the read counter RCNT, a signal EMPTY indicating the vacant state is output.

The registration flag circuit 560 is shown in detail in FIG. 24. In FIG. 24, 700 denotes an encoder, and R a register. The direction of the registered window number is stored in the register R after being encoded by the encoder 700 in accordance with the direction control signal DIR.

The address calculation circuit 580 outputs the window number to be informed to the adjacent control group and the upper address bits used for generation of the real address based on the window number read out from the FIFO 550. The detailed circuit is shown in FIGS. 25A to 25C.

In FIG. 25A, in the boundary of the window, the control group sends the window numbers (A+1) and (A–1) for the horizontal direction, and sends the window numbers (A+B) and (A–B) for the normal direction, where B denotes the number of the window data for the transverse direction when the virtual area is divided into the plural windows.

In FIG. 25B, the boundary of the window is distinguished by the boundary marks (E, W, N, S) 80. The value of each boundary mark is set by the controller 10 in the initial stage.
In FIG. 25C, ALU denote a calculation circuit, R80 to R82 denote registers, and S80 to S82 denote selectors. The calculation circuit ALU calculates any of the window numbers A, A×1, and A×2 in accordance with the boundary mark E, W, N, S shown in FIG. 25B. An address designation value ADD-DEG indicates a mode using the address transmitted from the controller 10 as an absolute address regardless of the present window number. When this mode is designated, the address designation value ADD-DEG is transmitted to the real address generation circuit 120 through the selectors S80 and S82.

FIG. 26 is a detailed block diagram of the real address generation circuit. In FIG. 26, R100 to R105 denote registers, S100 to 103 denote selectors, and OR denotes OR circuits. The input signals to this circuit are the relative address of the virtual PE transmitted from the controller 10, the upper address bits output from the address calculation circuit S80, and the adjacent window numbers input from the input circuit S50. The real address of the PE belonging to its own control group is generated by adding the relative address set in the register R100 to the upper address bits set in the register R101 as shown in FIG. 14B. As shown in FIG. 14B, in the upper eight bits, when the base address and the relative address overlap, one side is set to “0.” The real address is obtained by the logic OR calculation. In this case, the lower eight bits of the real address are the same as bits transmitted from the controller 10.

Further, to generate the real address for the adjacent pseudo PE, the window number of the adjacent PE is set in the registers R102 to R105. Further, the window numbers are controlled by the selectors S100 to S103 to be the address of the adjacent control group when loading (L), and to be the self-address when saving (S).

In this embodiment, although the multi-processor is constituted by lattice coupling, it is possible to constitute it by hyper-cubic coupling in accordance with the application.

The many features and advantages of the invention are apparent from the detailed specification and thus it is intended by the appended claims to cover all such features and advantages of the invention which fall within the true spirit and scope thereof. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly all suitable modifications and equivalents may be restored to as falling within the scope of the invention.

What we claim is:
1. A parallel computer system using a single instruction stream multiple data stream (SIMD) method, said parallel computer system having a controller and a plurality of processor elements, each of the processor elements having storage means for storing data to be processed and outputting output from the processor elements, the controller controlling operation of the processor elements, and said parallel computer system performing processing of the data based on a calculation control signal and a synchronization signal transmitted from the controller, said parallel computer system comprising:
   - data collection means, connected between the processor elements and the controller and arranged in a [binary] tree configuration having stages, for receiving the output data from the processor elements responsive to the synchronization signal generated by and received from the controller for performing a predetermined calculation based on the stages in the [binary] tree configuration, and for outputting calculated data to the controller; and
   - calculation control means, connected between said data collection means and the controller and arranged in series corresponding to each of the stages, for transmitting the calculation control signal from the controller to said data collection means based upon a pipe-line method to perform the predetermined calculation in said data collection means, wherein said data collection means comprises a plurality of gathering logic units connected to each other in the [binary] tree configuration having the stages, first gathering logic units of a first stage of the stages receives the output data from each of the processor elements and outputs first calculation data, second gathering logic units of a second stage of the stages receives the first calculation data obtained from the first stage, and the first calculation data obtained from the second stage is output to a final gathering logic unit of a first stage of the stages as second calculation data, the final calculation data obtained from the final gathering logic unit of the final stage responsive to said second calculation data is output to the controller, and wherein said calculation control means comprises a plurality of control registers each corresponding to one of the stages, each of the plurality of control registers connected in series to each other by the pipe-line method, and each of the plurality of control registers sequentially outputting the calculation control signal to each of the gathering logic units in the corresponding stage.
2. A parallel computer system as claimed in claim 1, wherein the calculation control signal is one of an AND calculation signal, an OR calculation signal, and a MAX/MIN calculation signal.
3. A parallel computer system as claimed in claim 2, wherein each of said gathering logic units comprises:
   - an OR calculation means for performing a logic OR calculation on the output data from each of the processor elements;
   - an AND calculation means for performing a logic AND calculation on the output data from each of the processor elements;
   - a MAX/MIN/ADD calculation means for obtaining one of a maximum value, a minimum value and an added value of the output data from each of the processor elements;
   - selector means for selecting one of said OR calculation means, AND calculation means and MAX/MIN/ADD calculation means, and for performing calculations in response to the calculation control signal.
4. A parallel computer system as claimed in claim 1, wherein each of said gathering logic units comprises:
   - an OR calculation means for performing a logic OR calculation on the output data from each of the processor elements;
   - an AND calculation means for performing a logic AND calculation on the output data from each of the processor elements;
   - a MAX/MIN/ADD calculation means for obtaining one of a maximum value, a minimum value, and an added value of the output data from each of the processor elements;
   - selector means for selecting one of said OR calculation means, AND calculation means and MAX/MIN/ADD calculation means, and for performing calculations in response to the calculation control signal.
5. A parallel computer system using a single instruction stream multiple data stream (SIMD) method, including processor elements and a controller generating control and synchronization signals, said parallel computer system comprising:

- data collection means, connected between the processor elements and the controller and arranged in a [binary] tree structure having stages, for receiving output data from the processor elements responsive to the synchronization signal received by the processor elements from the controller, for performing a predetermined calculation responsive to the control signal based on the stages in the [binary] tree structure, and for outputting calculated data to the controller; and
- calculation control means, connected between said data collection means and the controller and arranged in series corresponding to each of the stages, for transmitting the control signal received from the controller to said data collection means based upon a pipe-line process,

wherein said data collection means comprises a plurality of gathering logic units connected to each other in the [binary] tree structure having the stages, first gathering logic units of a first stage of the stages receives the output data from each of the processor elements and outputs first calculation data, second gathering logic units of a second stage of the stages receives the first calculation data obtained from the first stage, and the first calculation data obtained from the second stage is output to a final gathering logic unit of a final stage of the stages as second calculation data, and final calculation data obtained from the final gathering logic unit of the final stage responsive to said second calculation data is output to the controller, and

wherein said calculation control means comprises a plurality of control registers each corresponding to one of the stages, each of the plurality of control registers connected in series to each other by the pipe-line method, and each of the plurality of control register sequentially outputting the control signal to each of the gather logic units in the corresponding stage.

6. A parallel computer system as claimed in claim 5, wherein the calculation control signal is one of an AND calculation signal, an OR calculation signal, and a MAX/MIN calculation signal.

7. A parallel computer system as claimed in claim 5, wherein each of said gathering logic units comprises:

- an OR calculation means for performing a logic OR calculation on the output data from each of the processor elements;
- an AND calculation means for performing a logic AND calculation on the output data from each of the processor elements;
- a MAX/MIN/ADD calculation means for obtaining one of a maximum value, a minimum value and an added value of the output data from each of the processor elements; and
- selector means for selecting one of said OR calculation means, AND calculations in response to the calculation and for performing calculations in response to the calculation control signal.

8. A method of calculation processing, comprising:

- processing input data in parallel in plural processing units to produce parallel output data;
- combining the parallel output data using a tree combination of tree stages to produce calculated output data; and
- controlling said processing and controlling said tree stage combining at each stage to produce a predetermined calculated output data result.

9. A method of calculation processing as recited in claim 8, wherein said combining of the parallel output data is performed logically.

10. A method of calculation processing as recited in claim 8, wherein said controlling of said processing and said combining is performed in pipeline stages.

11. An apparatus, comprising:

- parallel processing units processing input data in parallel to produce parallel output data;
- data combination units connected to said parallel processing units and combining the parallel output data using a tree combination having combining control at each stage to produce calculated output data; and
- a controller connected to said data combination units and said parallel processing units and controlling said processing and said combining to produce a predetermined calculated output data result.

* * * * *