SPARTH: super-precision parallel arithmetic processor with dynamic changing of the operand length
A. Vazhenin

Department of Software of High-Performance Computing Systems, Computing Center, Siberian Division of the Russian Academy of Sciences, Novosibirsk, Lavrentiev ave., 6, 630090, Russia

ABSTRACT

An efficient way to achieve the high accuracy of the results of computations is to increase the operands capacity. The best results in terms of the problem solution rate and effectiveness of memory using can be reached in the case when a computer system provides the possibility of dynamic capacity control and parallel data processing. The suggested programming system presents an effective tool for Super-precision Parallel ARithmetic computations (SPARTH-computations). It is developed for a STARAN-like computing system and oriented to solve a large set of vector and matrix operations. From the user’s viewpoint the system represents a parallel vector processor with programmable word length called SPARTH-processor. It can provide also the accuracy control of computations directly during solution of the problem. This paper describes the main features of the SPARTH-processor architecture and new parallel algorithms of accurate computing of dot products and polynomials, multiplication of matrices with multidigital elements, and iterative algebraic linear equations solving with dynamic control of resulting accuracy. The results of an estimation of the SPARTH-processor accuracy and its performance are also presented.

INTRODUCTION

One of the important factors determining the accuracy of data processing results is rounding in arithmetic operations. The necessity of rounding is caused by the fixed and relatively small length of operands in general purpose computers. The decrease of errors may be achieved by means of using multi-precision arithmetic. Super-long operands may be processed either
Applications of Supercomputers in Engineering

by using of specialized coprocessors [8],[13] or by program implementation of multiprecision arithmetic algorithms in terms of the basic computer architecture [1], [7]. However, the program implementation or micro-program interpretation of high accuracy computations leads (in usual computing systems) to a fast decrease of problem solution rate and non-effective use of memory.

The development of current integral technology and computer science allows to design parallel systems functioning with varying operand length. This led to computer systems solving problems at an accuracy given before calculations or an accuracy provided by system resources and known to the user after the computation terminates. In [5] a program package for multiple precision integer arithmetic and numeric computations for CRAY-2 is described. These programs implement multiple precision arithmetic operations employing the pipeline principle. In [9] the language for high-accuracy computations and the way of implementation of this language in a multiprocessor system are proposed.

From the viewpoint of effective performance and programming of high accuracy computations, the prospective computing systems may be the SIMD parallel systems with vertical processing. Their distinctive features combining the possibilities of location and parallel processing of operands of arbitrary length (up to several thousands bits), programmability of data formats, data masking, etc., allow to consider an effective means to implement Super-precision Parallel ARiTHmetic computations (SPARTH-computations).

The vertical processing is based on the word parallelism. Bit slices of processed array are extracted from memory in a regular way, then they are input to the registers of operating unit, where they are processed by appropriate logic circuits. Such processing is called ”bit-serial” [4]. Similar devices are also called systems with ”fine-grained” structure. The main feature of any of these system is the presence of a large number of one-bit processing elements (PE) operating in parallel, each having one-bit local memory and performing bit-serial processing on its contents. The well-known systems of this type are STARAN [2], DAP [10], MPP [3], CM [15], LUCAS [6], etc. Their performance ranges up to several billions of bit operations per second.

The present paper deals with the programming system of SPARTH-computations for STARAN-like architectures. It is oriented for problems containing many vector and matrix operations and allows to control of the computation accuracy during solving these problems.
PRINCIPAL FEATURES OF THE SPARTH-PROCESSOR

SPARTH-architecture

Fig. 1 shows the SPARTH-processor architecture implemented within the basic STARAN-architecture. The main elements of SPARTH-processor are:

- Vector registers $V_{R_0} - V_{R_{v-1}}$ intended for the location of super-digital vector operands;
- High-precision parallel summator (HPS) containing fields $V_{S_0} - V_{S_3}$ and having the capacity necessary for the performance of arithmetic operations without rounding;
- Scalar registers or scalars $S$ where scalar operands are located;
- Operational registers $X, Y, M$;
- Index registers or indices $I$ which are unsigned integers and intended for storage of constants defining the number of loop iterations, modes of access to vector registers, etc.;
- Registers for temporal storage of masks $R_{M_0} - R_{M_{l-1}}$ located in the special field of the multidimensional access (MDA) memory and intended for storing the masks and bit slices resulting from performance of parallel vector operations (overflow, search operations, etc.).

All parallel operations are performed for unmasked elements of vector registers. The masks for these instructions should be loaded to the $M$-register. Functions of the $X$- and $Y$-registers are similar to their assignment in STARAN. They provide processing of bit slices. A bit slice resulting of the parallel operation performance is located in the $F$-register. It may be written either in the $R_{M}$-register or in the $M$-register, or processed in operational registers $X$ and $Y$.

Computations in the SPARTH-processor may be performed in two modes: with the fixed or dynamic accuracy. The first mode is characterized by the constant capacity of operands in computations. In this case, the number $v$ of vector registers available is determined by required capacity

$$v = \left\lfloor \frac{s - 4n - l}{n} \right\rfloor,$$  

Figure 1. The SPARTH-processor architecture
where \( n \) is a ordered capacity (bits), \( l \) - number of \( RM \)-registers, and \( s \) - size of local memory. The user is able to choose between the problem solution rate, the amount of processed data and required accuracy.

In the dynamic accuracy mode, the operand capacity may alter in the interval defined by the user. The number of available vector registers is formed according to the maximum ordered capacity value (see Equation 1). Computations may be started with relatively small capacity of operands. If needed, the SPARTH-processor may be switched to a next capacity limit by means of precision control procedures.

In the SPARTH-processor three types of data are used: integer data type; fixed-point type format (without integer part); real type format (numbers with integer and fractional parts).

**SPARTH-instructions**

Processor instructions provide effective interaction of subsystems and execution of high-precision parallel computations. Arithmetic operations are executed in two stages. At first, the exact result (without rounding) is formed in HPS, then it is stored into destination registers using the rounding operations for multiplication and division. In the dynamic accuracy mode, the data located in HPS may be stored in vector registers without rounding.

In Fig. 1 \( m \) denotes the number of PE’s. All \( m \) components of vectors may be performed in parallel. Therefore, the parallel addition and subtraction have an arithmetic operation count of \( O_A(1) \) and a bit operation count of \( O_B(n) \) (for \( n \)-digital numbers). Operation counts of both parallel multiplication and division are \( O_A(1) \) and \( O_B(n^2) \). The accurate sum of vector elements is executed by recursive doubling in groups. Each of these groups may contain \( 2, 4, 8, \cdots, 2^i, \cdots, m \) components. The sum is computed for each group in parallel and have operation counts of \( O_A(\log_2 N) \) and \( O_B(n \log_2 N) \), where \( N = 2^i \).

The change-over of capacity limits from \( n \) to \( 2n \) bits has a bit operation count of \( O_B(vn) \).

Data transmission instructions allow the user to assign different operations of data exchange between subsystems of SPARTH-processor using the properties of the FLIP interconnection network [2]. They support different procedures of the parallel transmission of unmasked elements of vector registers, and selective communication of the data chosen by means of index registers. For parallel transfer operations, elements may be interchanged. There are three type of permutations: mirror, cyclic shift and mixed permutation which use the both previous types (see [12] for details). Data
transmission instructions have operation counts of $O_A(1)$ and $O_B(n)$.

Special procedures

These procedures execute structural transformations of vectors, and provide special types of computations. This allows effective mapping of algorithms on the SPARTH-processor architecture.

Expansion and compression of vectors

As indicated above, the sum of vector elements is computed for $N = 2^i$. Special procedures was developed to provide the processing of vectors of arbitrary $N$.

**Definition 1.** Let $X$ contains $k$ groups of $N$ components each:

$$X = \{x_0, x_1, \ldots, x_{N-1}, \ldots, x_{0}^{k-1}, x_{1}^{k-1}, \ldots, x_{N-1}^{k-1}\}.$$  

A procedure forming for $m \geq N \cdot k$ a vector

$$\tilde{X} = \{x_{0}, \ldots, x_{N-1}, c, \ldots, c, \ldots, x_{0}^{k-1}, \ldots, x_{N-1}^{k-1}, c, \ldots, c\},$$

where $N_1 = 2^{[\log_2 N]}$ and, $c$ is a constant, we call vector-expansion.

To calculate a sum, it is necessary to set $c = 0$.

**Definition 2.** Let $X$ contains $k$ groups of $N_1 = 2^i$ each. A procedure forming for $m \geq N_1 \cdot k$ a vector

$$\tilde{X} = \{x_{0}, x_{1}, \ldots, x_{N-1}, \ldots, x_{0}^{k-1}, x_{1}^{k-1}, \ldots, x_{N-1}^{k-1}, x, \ldots\},$$

where $2^{i-1} < N < N_1$ and $x, \ldots$ is a "tail" of length $k(N_1 - N)$, we call vector-compression.

The operation counts of both vector-expansion and vector-compression procedures in the SPARTH-processor are $O_A([\log_2 N])$ for arithmetic operations, and $O_B(n[\log_2 N])$ for bit operations.

Prefix and suffix procedures

These procedures may be used as a part of the operational basis for developing of many parallel numerical and nonnumerical algorithms (evaluation of polynomials, trancedent functions, factorials, etc.).

**Definition 3.** Let $C$ contains $p$ components $C = \{c_{p-1}, \ldots, c_1, c_0\}$. The following procedures forming a vector $\tilde{C} = \{\tilde{c}_{p-1}, \ldots, \tilde{c}_1, \tilde{c}_0\}$ we call:

- prefix sum, if $\tilde{c}_j = \sum_{i=0}^{j} c_i$;
- suffix sum, if $\tilde{c}_j = \sum_{i=0}^{j} c_{p-1-i}$.
- prefix product, if $\hat{c}_j = \prod_{i=0}^{j} c_i$;
- suffix product, if $\hat{c}_j = \prod_{i=0}^{j} c_{p-1-i}$.

The computation of prefix and suffix operations in the SPARTH-processor is executed by parallel data transmission with cyclic shifts of vector and masks, and parallel addition or multiplication. The number of $\hat{c}_j$ is doubled in each step. Therefore, the operation counts of prefix and suffix procedures are $O_A(\log_2 p)$ and $O_B(n \log_2 p)$ (for sums), or $O_A(\log_2 p)$ and $O_B(n^2 \log_2 p)$ (for products). In the next section we demonstrate the parallel polynomial evaluation using prefix product.

**APPLIED ALGORITHMS AND NUMERICAL EXPERIMENTS**

In this section examples of solving of some typical problems are shown. The results of estimation of SPARTH-processor accuracy and performance are presented. Computations were implemented in STARAN-like system with following characteristics: $m = 256$, execution times of addition and multiplication are $t_a \approx 6n\tau$ and $t_m \approx 6n^2\tau$ correspondingly, where $\tau \approx 200\text{ns}$ is the average execution time of each parallel bit operation.

Accurate scalar products

A fundamental algorithm in regard to basic numerical methods is the one for the computation of the scalar product (dot product) of vectors $\mathbf{X} \equiv [x_i]$ and $\mathbf{Y} \equiv [y_i]$:

$$P = \mathbf{X} \cdot \mathbf{Y} = \sum_{i=0}^{N-1} x_i \cdot y_i.$$

The computation of this formula can be made in the SPARTH-processor employing the above mentioned instructions of multiplication of vectors $\mathbf{X}$ and $\mathbf{Y}$, and sum of vector elements.

**Theorem 1.** The operation counts of scalar product computation in a SPARTH-processor are $O_A(\log_2 N)$ and $O_B(n^2 + n \cdot \log_2 N)$. 

The flexible switching of capacity limits in the SPARTH-processor allows an automatic adaptation to the range of data values. As indicated in the SPARTH-program given below, the algorithm of exact scalar product is implemented in following stages:

- parallel conversion of input floating-point numbers with automatic selection of capacity needed for exact representation of these numbers;
- vector-expansion of $\mathbf{X}$ and $\mathbf{Y}$;
- calculation of dot products in parallel;
- vector-compression of results and transformation to the floating-point format.
To estimate the accuracy of scalar product evaluating in SPARTH-processor we have used the "hard" input data from [11]. Examples of such data are shown in Table 1. In [11] test results of parallel systems SIEMENS/Fujitsu VP400-EX and CRAY-2, and high-accuracy arithmetic subroutine library ACRITH on IBM-4381 [7] are also described.

The results from Table 2 show that the accuracy of dot products computations in SPARTH-processor is similar to ACRITH. However, SPARTH
Multiplication of matrices with multidigital elements

Matrix multiplication is computation intensive, because multiplying two $N$ by $N$ matrices takes $O_A(N^3)$ arithmetic operations and $O_B(N^3(n^2 + n))$ bit operations. Studying parallel matrix multiplication has the advantage of exposing the tradeoff among various degrees of parallelism, performance, and communication patterns on a simple and easily understood problem. Our goal is to examine as well the influence of capacity of matrix elements on the SPARTH-processor performance and speedup, and to consider the possibilities of fast accurate matrix multiplication using the dynamical capacity mode of computation.

The usual specification of matrix multiplication is

$$C = AB \equiv [a_{ij}][b_{jk}] = [c_{ik}],$$

where $c_{ik} = \sum_{j=0}^{N-1} a_{ij}b_{jk}, 0 \leq i, j \leq N - 1$.

There are different algorithms of matrix multiplication depending on the relation between $N$ and $m$ [14]. Here we examine the parallel algorithm for $m \geq N^2$.

**Theorem 2.** The operation counts of multiplication of two $N$ by $N$ matrices containing $n$-digital elements are $O_A(N[\log_2 N])$ and $O_B(N^2 + n N[\log_2 N])$ for SPARTH-processor having $m \geq N^2$ PE’s.
PROOF. As shown in Fig. 2 (for \( N = 4 \)), the matrices \( A \) and \( B \) are placed in the corresponding vector registers by rows for \( A \) and by columns for \( B \). By calculating \( N \) scalar products in groups of \( N \) components each in parallel we produce \( N \) elements of \( C \) which are loaded into the destination vector register taking into account the masks (RM-register). Next is the exchange of columns of \( B \) and RM-register using the cyclic shift permutation. If \( N \neq 2^i \), then it is necessary to implement vector-expansion of products before calculating of sum, and vector-compression of results after summing. The operation counts of each iteration are \( O_A(\log_2 N) \) and \( O_B(n^2 + n\log_2 N) \), the total number of such iterations is \( N \). Therefore, the total operation counts of matrix multiplication are \( O_A(N \log_2 N) \) and \( O_B(Nn^2 + nN\log_2 N) \).

<table>
<thead>
<tr>
<th>VR0</th>
<th>VR1</th>
<th>HPS</th>
<th>HPS</th>
<th>RM0</th>
<th>VR3</th>
<th>VR1</th>
<th>RM0</th>
<th>VR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>0</td>
<td>c00</td>
<td>b10</td>
<td>0</td>
<td>c00</td>
</tr>
<tr>
<td>10</td>
<td>01</td>
<td>a10b01</td>
<td>0</td>
<td>*</td>
<td>b11</td>
<td>1</td>
<td>c10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>02</td>
<td>a20b02</td>
<td>0</td>
<td>*</td>
<td>b12</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>03</td>
<td>a30b03</td>
<td>0</td>
<td>*</td>
<td>b13</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>10</td>
<td>a04b10</td>
<td>c11</td>
<td>0</td>
<td>b20</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>a11b11</td>
<td>c11</td>
<td>c11</td>
<td>b21</td>
<td>0</td>
<td>c11</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>12</td>
<td>a21b12</td>
<td>c11</td>
<td>0</td>
<td>b22</td>
<td>1</td>
<td>c21</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>13</td>
<td>a31b13</td>
<td>c11</td>
<td>0</td>
<td>b23</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>20</td>
<td>a02b20</td>
<td>c22</td>
<td>0</td>
<td>b30</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>a12b21</td>
<td>c22</td>
<td>0</td>
<td>b31</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>a22b22</td>
<td>c22</td>
<td>c22</td>
<td>b32</td>
<td>0</td>
<td>c22</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>23</td>
<td>a32b23</td>
<td>c22</td>
<td>0</td>
<td>b33</td>
<td>1</td>
<td>c32</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>30</td>
<td>a03b30</td>
<td>c33</td>
<td>0</td>
<td>b00</td>
<td>1</td>
<td>c03</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>31</td>
<td>a13b31</td>
<td>c33</td>
<td>0</td>
<td>b01</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>32</td>
<td>a23b32</td>
<td>c33</td>
<td>0</td>
<td>b02</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>33</td>
<td>a33b33</td>
<td>c33</td>
<td>c33</td>
<td>b03</td>
<td>0</td>
<td>c33</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Matrix multiplication in the SPARTH-processor

Theorem 3. The relative speedup of multiplication of two \( N \) by \( N \) matrices containing \( n \)-digital elements is \( O\left(\frac{N^2n}{n+\log_2 N}\right) \) for SPARTH-processor having \( m \geq N^2 \) PE’s.

The results of measurements of relative speedup show (Fig.3), that it is growing with increasing of operand length. Hence, the SPARTH-processor may provide effective high-accuracy computations.

The SPARTH-processor may process \( \mu = \frac{m}{2^i\log_2 i} \) pairs of matrices simultaneously. That supports the maximal level of parallelism. Fig.4 contains the results of measurements of maximal SPARTH-processor performance.
The fast accurate matrix multiplication may be implemented using the dynamic accuracy mode of computations. In this case, the accurate intermediate scalar products having a capacity of $n_1 = n + \lceil \log_2 N \rceil$ bits are formed in HPS. Then they are stored in the destination register without rounding. After calculating of all elements of $C$, the SPARTH-processor is switched to next capacity limit. Compared to fixed accuracy mode, this allows to increase the solution rate up to 3.5-times.

### Polynomial evaluation

Usually the evaluation of a polynomial

$$P(x) = a_{p-1}x^{p-1} + \cdots + a_1x + a_0 = \sum_{i=0}^{p-1} a_i x^i$$

is done via Horner's scheme. This leads to a linear first order recurrence for which a vectorization is possible only with the help of an expansion method like recursive doubling or cyclic reduction.

We use another simple and fast method to evaluate a polynomial. It is to compute $P(x) = \mathbf{A} \cdot \mathbf{X}$ as the dot product of $\mathbf{A} = \{a_{p-1}, \ldots, a_1, a_0\}$ and $\mathbf{X} = \{x^{p-1}, \ldots, x, 1\}$. The vector $\mathbf{X}$ may be computed using the prefix product of vector

$$\dot{\mathbf{X}} = \{x, x, \ldots, x, 1\}.$$

Theorem 4. The operation counts of polynomial evaluation in a SPARTH-processor are $O_A(\log_2 p)$ and $O_B(n^2 \log_2 p + n^2 + n \log_2 p)$.

Proof. The polynomial evaluation may be implemented in three stages: the forming of $\hat{X}$ in $O_A(1)$ and $O_B(n)$ time steps loading a scalar $x$ into the vector register, calculation of $\hat{X}$ by prefix product and dot product of $A$ and $X$. Therefore, total operation counts of polynomial evaluation are $O_A(\log_2 p)$ and $O_B(n^2 \log_2 p + n^2 + n \log_2 p)$. □

Theorem 5. The relative speedup of polynomial evaluation in SPARTH-processor is $O\left(\frac{p}{\log_2 p + 1}\right)$.

Proof. The polynomial evaluation in sequential computers is done via Horner's scheme in $p$ multiplications and $p$ sums. Therefore, the sequential bit operation count is $O_B(pn(n + 1))$. Thus, the ratio of sequential and parallel bit operation counts is $O\left(\frac{p}{\log_2 p + 1}\right)$, because $\log_2 p = \lfloor \log_2 p \rfloor$ for $p = 2^i$. If $p \neq 2^i$, the computation is implemented in $p_1 = \lfloor \log_2 p \rfloor + 1$ using the vector-expansion. □

Corollary 1. The maximal relative speedup of polynomial evaluation in SPARTH-processor is $O\left(\frac{mp}{2^{\lfloor \log_2 p \rfloor} + \left\lfloor \log_2 p \right\rfloor + 1}\right)$.
Applications of Supercomputers in Engineering

**Proof.** A SPARTH-processor can process $N_1 = \frac{m}{2^{[\log_2 p]}}$ simultaneously in groups of $2^{[\log_2 p]}$ components each using the properties of the FLIP interconnection network. If all $m$ processing channels are used, then the maximal relative speedup is $O\left(\frac{mp}{2^{[\log_2 p]}([\log_2 p] + 1)}\right)$. □

To estimate the accuracy of polynomial evaluation we have used the "hard" input data from [11]. The coefficients were defined as

$$a_i = \begin{cases} 
-16^7, & \text{for } i = 8 \\
16^4, & \text{for } i = 11 \\
16^i, & \text{in other cases}
\end{cases}$$

The input data are $x_j = 16 + j \cdot 16^7$, where $(j = -6, -5, \ldots, -1, 0, 1, \ldots, 4)$.

Polynomials were computed simultaneously for all values $x_j$ using a procedure similar to the calculation of dot products. It is an automatic adaptation to the range of data values. As shown in Table 3, the polynomial evaluation in SPARTH-processor allows calculation of absolutely exact results.

<table>
<thead>
<tr>
<th>$j$</th>
<th>$P_j$ (scalar) VP400-EX</th>
<th>$P_j$ (vector) VP400-EX</th>
<th>$P_j$ (Horner) VP400-EX</th>
<th>$P_j$ (ACRITH) VP400-EX</th>
<th>$P_j$ (SPARTH) VP400-EX</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6</td>
<td>-4831838152</td>
<td>-4831838140</td>
<td>-4831838162.0</td>
<td>-4831838139.25</td>
<td>-4831838133.25</td>
</tr>
<tr>
<td>-5</td>
<td>-4026531800</td>
<td>-4026531788</td>
<td>-4026531810.5</td>
<td>-4026531789.81</td>
<td>-4026531783.81</td>
</tr>
<tr>
<td>-4</td>
<td>-3221225448</td>
<td>-3221225436</td>
<td>-3221225456.0</td>
<td>-3221225437.00</td>
<td>-3221225431.00</td>
</tr>
<tr>
<td>-3</td>
<td>-2415919096</td>
<td>-2415919084</td>
<td>-2415919098.5</td>
<td>-2415919080.81</td>
<td>-2415919074.81</td>
</tr>
<tr>
<td>-2</td>
<td>-1610612728</td>
<td>-1610612716</td>
<td>-1610612738.0</td>
<td>-1610612721.25</td>
<td>-1610612715.25</td>
</tr>
<tr>
<td>-1</td>
<td>-805306360</td>
<td>-805306348</td>
<td>-805306374.5</td>
<td>-805306358.31</td>
<td>-805306352.31</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>4</td>
<td>8.0</td>
<td>8.00</td>
<td>14.00</td>
</tr>
<tr>
<td>1</td>
<td>805306376</td>
<td>805306372</td>
<td>805306377.5</td>
<td>805306377.68</td>
<td>805306383.67</td>
</tr>
<tr>
<td>2</td>
<td>1610612744</td>
<td>1610612740</td>
<td>1610612750.0</td>
<td>1610612750.75</td>
<td>1610612756.75</td>
</tr>
<tr>
<td>3</td>
<td>2415919112</td>
<td>2415919108</td>
<td>2415919125.5</td>
<td>2415919127.18</td>
<td>2415919133.18</td>
</tr>
<tr>
<td>4</td>
<td>3221225448</td>
<td>3221225476</td>
<td>3221225504.0</td>
<td>3221225507.00</td>
<td>3221225513.00</td>
</tr>
</tbody>
</table>

Iterative algebraic linear equations solving

A well-known method for solving a system of linear equations $AX = B$ is the iterative method $X^{k+1} = DX^k + B$, where $D = I - A$. Usually, the condition of computation termination is

$$|X^{k+1} - X^k| \leq \epsilon,$$

where $\epsilon > 0$ is a parameter defining the accuracy of solution.

**Theorem 6.** The operation counts of iterative solving of a system containing $N$ linear equations in the SPARTH-processor with $m \geq N^2$ PE's are $O_A([\log_2 N])$ and $O_B(n^2 + n[\log_2 N])$ for each iteration.
Proof. Matrix $D$ is placed into a vector register by rows. Vectors $X^k$ and $B$ are placed in corresponding vector registers, and multiplied by groups of $N$ elements each. The multiplication of $D \times X^k$ is implemented in parallel using accurate scalar product in groups of $N$ elements each. The vector $X^{k+1}$ is computed by addition of scalar products with $B$. The operations counts of the test of condition (Equation 2) are $O_A(1)$ and $O_B(n)$. Therefore, the total operation counts are $O_A\left(\frac{N^2}{n + \log_2 N}\right)$ and $O_B\left(n^2 + n\log_2 N\right)$.

Theorem 7. The relative speedup of iterative solving of a system containing $N$ linear equations in the SPARTH-processor with $m \geq N^2$ PE’s is $O\left(\frac{N^2}{n + \log_2 N}\right)$ for each iteration.

Proof. To calculate the vector $X^{k+1}$ in sequential computers, it is necessary to execute $N$ scalar products, vector addition, and testing of condition (Equation 2), which requires $O_B(N^2(n^2 + n))$, $O_B(Nn)$ and $O_B(Nn)$, correspondingly. Thus, the sequential bit operation count of each iteration is $O_B(N^2(n^2 + n) + Nn)$. The desired result is obtained by division of sequential and parallel bit operation counts.

The accurate calculation of each iteration may decrease the necessary number of iterations. Table 4 contains the results of numerical experiments. The "hard" input data and the results for comparison were taken from [11]. Coefficients of matrix $A \equiv [a_{ij}]$ were defined as

$$a_{ij} = \begin{cases} \frac{15}{16}, & \text{for } i = j \\ -\frac{1}{16}, & \text{for } i \neq j \end{cases} \quad (i, j = 0, \ldots, 11).$$

The vector $B$ was $B = \{16^{15}, 1, 1, -16^{15}, 1, 1, 1, 1, 1, 1, 1\}$. The accurate solution of this system is the vector

$$X = \{16^{15} + 2, 3, 1, -16^{15} + 2, 3, 3, 3, 3, 3, 3, 3\}.$$

As shown in Table 4, the SPARTH-processor provides more accurate calculations than computers with usual data formats (64-bits floating-point numbers) in spite of the lesser number of implemented iterations.

A fast algorithm with multistep refinement of results was developed (Fig. 5). The speedup is achieved by using of small operand length at early stages of computation. In other words, rough approximations of results are calculated at first. Then, they are improved by switching to next capacity limits. In this case, the parameter determining required precision is a vector $E = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_j, \ldots, \epsilon_l\}$. Each $\epsilon_j$ defines the accuracy at corresponding capacity limit. When the condition of terminating of current computation step is reached, the SPARTH-processor is switched to next capacity limit, and computations are implemented with new $\epsilon_j$. 
Table 4. Results of iterative solving of linear equations

<table>
<thead>
<tr>
<th>i</th>
<th>$X_i$ (scalar)</th>
<th>$X_i$ (vector)</th>
<th>$X_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VP400-EX</td>
<td>VP400-EX</td>
<td>SPARTH</td>
</tr>
<tr>
<td>0</td>
<td>1152921504606846980</td>
<td>1152921504606846980</td>
<td>1152921504606846978</td>
</tr>
<tr>
<td>1</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>2</td>
<td>-0.66666666666666668</td>
<td>-0.66666666666666668</td>
<td>0.973273077979687600</td>
</tr>
<tr>
<td>3</td>
<td>-1.152921504606846980</td>
<td>-1.152921504606846980</td>
<td>-1.1529215046068469740</td>
</tr>
<tr>
<td>4</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>5</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>6</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>7</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>8</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>9</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>10</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
<tr>
<td>11</td>
<td>1.999999999999999978</td>
<td>1.33333333333333326</td>
<td>2.973273077979687600</td>
</tr>
</tbody>
</table>

Figure 5. Multistage iterative solving of linear equations
The speedup value depends on the relation between initial and final operand lengths defined by required precision and the features of a concrete system solved. For example, if the initial length of \( n_0 \) and resulting precision of \( \epsilon_t \) are \( n_0 = 32 \) bits and \( \epsilon_t = 10^{-64} \), then the speedup value is about 5.81 for the system given above. If \( n_0 = 64 \) bits and \( \epsilon_t = 10^{-32} \), it is about 2.22. This confirms the effective implementation of high-accuracy computations in the SPARTH-processor.

CONCLUSION

The comparison with known dedicated programming systems for high-accuracy computations on sequential computers shows that SPARTH-processor ensures similar accuracy of results. Moreover, it provides very high performance due to effective use of massively parallelism. It may be considered also as an intermediate language for the translation from high-level parallel scientific languages (FORTRAN-XCS, PASCAL-XCS, C-XCS, etc.) It may simplify the structure of such compilers, because the SPARTH-instructions are relatively large vector operations. This approach may be used as well for other massively parallel systems (DAP, MPP, CM, etc.). It can also lead to developing of new computer architectures with overcoming of rounding errors.

REFERENCES


136 Applications of Supercomputers in Engineering


