EURASIAN JOURNAL OF MATHEMATICAL AND COMPUTER APPLICATIONS
ISSN 2306–6172
Volume 2, Issue 2(2014) 92–101

SOME REGULARITIES AND OBJECTIVE LIMITATIONS OF IMPLEMENTING SEMANTIC PROCESSING ALGORITHMS ON COMPUTING SYSTEMS WITH MASSIVE PARALLELISM

N.L. Verenik, Y.N. Seitkulov, A.I. Girel, M.M. Tatur

Abstract In the article hardware implementation approaches when building applied intelligent systems is briefly described, the problem to achieve high efficiency by the low cost is presented. Authors propose to develop semantic problem-oriented ASIP, whose architecture, data format and instruction set are presented in the paper. As the base architecture SIMD-architecture is used. Basic principles of functioning are considered. Several architecture limitations and objective regularities are given. Usage example of processor for solution of the pathfinding in graph problem is briefly presented.

Key words: semantic information processing; semantic network; ASIP; massive parallel processing.

AMS Mathematics Subject Classification: 68R10

1 Introduction

One of the most promising approaches to formalize the functioning of the custom intelligent system is the use of semantic networks by means of which knowledge of the system (knowledge base of the system), as well as intelligent information processing algorithms (used by this system) may be represented. Basically the semantic network is a certain graph structure, the elements of which are given an additional meaning. Computer system, on the basis of which a semantic network operates, is usually a so-called graph-dynamic machine [1], it is a software or hardware system, the internal state of which is represented by a graph. Information processing in such a system is interpreted as a graph-dynamic process, i.e. the process of change of the system internal graph structure, which may include not only changes in the internal state of graph elements, but also the change of graph configuration (adding or removing vertices and/or arcs in the graph).

In practice, in the process of the application intelligent system development the semantic network programming model is usually used, this model is implemented on a general-purpose computer system (PC), which is obviously significantly reduces the overall cost of the system and often is the best solution. However the use of general-purpose processors has a number of scalability limitations [2], [3]. In the case of implementation of supercomplex intelligent systems, which have an extremely high demands for performance or maximum knowledge base value, supercomputers can be used [4], [5], but this is often impossible because of the high cost of such a system. Moreover mass and dimensions parameters of supercomputers, increased maintenance charge (al-

location of special premises, cooling , etc.) and the high complexity of the attendants can be related as obvious disadvantages of them.

Another well-known solution is the use of parallel general-purpose systems, such as the GPU [6], clusters, cloud computing, which allows to achieve much better performance versus conventional general-purpose processors while maintaining a relatively low cost of the entire system. One of the drawbacks of such systems is the fact that they were originally designed to solve another kind of problems rather than the intelligent treatment tasks.

Alternative solution proposed by the authors of this work is to develop a problem-oriented ASIP [7]. Because of the initial focus on the solution of semantic processing problems it is possible to achieve better efficiency in comparison with one-stop solutions (general-purpose processors and general-purpose parallel processors) with maintaining a low total cost. There is a brief description of the developed architecture, instructions set and processor data format in the article, it also provides the information about the objective regularities and limitations of the processor implementation.

2 Problem-oriented semantic processor

The work [8] illustrates how the problems of semantic analysis can be reduced to the problems in graph theory, in particular, how an custom semantic network can be represented by a graph of a regular structure. In turn, the problems of graph theory mostly relate to combinatorial problems, which in practice can only be solved by the introduction of a number of restrictions in the initial problem situation. Such restrictions are imposed until the time to solve the problem reaches the necessary maximum permitted value. As a result, the deduced solution is meaningful only in the context of application problem for which it was designed.

At the same time, the development of a problem-oriented processor requires a certain period of time, and consequently material costs. If such a project is focused only on one particular problem, it will never reach economic efficiency. Therefore, from the very beginning it is necessary to lay a certain unification in the development of the processor architecture, it means to direct processor to solve a certain class of intricate problems [7].

The authors propose to implement hardware support for basic operations used in the graph theory algorithms, it means to perform efficiently search operations of graph elements and operations at sets of graph elements. Actually, the processor of the architecture proposed can be regarded as a complex associative memory.

2.1 Processor architecture

Figure 1 illustrates a block diagram of the processor. As the base architecture SIMD-architecture (Flynn's taxonomy) is used. The obvious advantages of this architecture are simplicity of connections between modules, the lowest hardware costs, a high level of modularity and, what is important, the scalability. So disadvantage is the large number of connections in the circuit and caused by it low reliability (in case of damage of one of the data buses), both of which is not significant in case of implementation on PLD. The

main problem is limited throughput (in particular in the time of data reading). On the

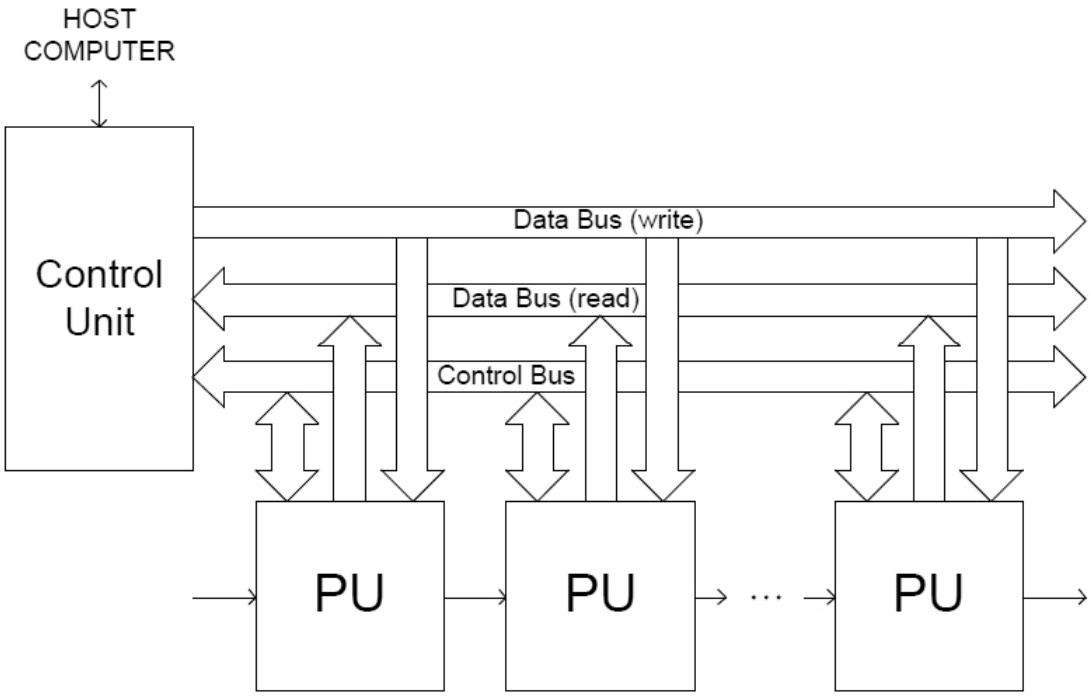

Figure 1: Processor block diagram

scheme the processor is represented by a single common control unit (CU), which reads and decodes the original program instruction sequence (single instruction stream), and by a set of processor units (PU), on which then the instructions are performed. CU interacts with array of PU through global data buses of concurrently three types:

- Data Bus to read - the bus through which the current instruction is transmitted simultaneously on all PU;
- Data Bus to write - the bus through which the serial data reading of processing elements (bus with the time division) is performed;
- Control Bus - the set of control signals.

Each PU has its own local private memory, only this particular PU is responsible for the interaction with it. PU are ganged among themselves by unidirectional local connections by means of which there is a possibility of high-priority reading data from the memory. Shared processor memory consists of the sum total of local memory of all the PU.

single cell	operation type	M_A, mask-address	D_A, data-address	M_D, mask-data	D_D, data-data

Figure 2: Processor instruction structure

3.2 Data representation and addressing mechanism

Let's examine the general mechanism of interaction with the processor memory. The system of processor instructions is represented by only one command (see Figure 2), which is used for both reading and writing data into the processor.

To perform the addressing (the selection of one or more elements of the processor memory for performing the action), an instruction includes a bit mask M_A and bit field D_A, their dimension coincides with the processor word. If there is a data point D_0 in the cell, then it is considered addressed (selected, active), when all the bits D_A and D_0, marked by the mask M_A are respectively equal, i.e. $D_{Ai} = D_{0i}$ for $\forall i, i = \overline{0, b-1}$ (where b is processor bitwise), so that $M_{Ai} = 1$.

It is often required (eg., in case of read operation) to apply the action not to more than one memory element. To select this mode the bit 'single cell' is used. In this case if more than one cell has been addressed, the choice of the cell for making a record will be carried out according to the scheme of priorities. The priority itself is determined by the sequence order in the scheme. For the simplicity, the cell that holds the most left position in the line of PU (see Figure 1) can be considered as activated one.

The action of writing data into the cell's memory also uses access through a mask, in particular M_D value defines the bits of the cell, which will be changed depending on the current bit value according to some function $f(D_{0i}, D_{Di}) \Rightarrow D_{0i}$ for $\forall i, i = \overline{0, b-1}$, such as $M_{Di} = 1$. The selection of function is determined by the field 'operation type'. At this stage simple assignment is expected to be implemented, and at least the basic functions of binary logic (AND, OR, NOT).

Figure 3 illustrates a block diagram of PE. The structure of PE includes local memory and combinational circuit performing the following operations:

- serial memory comparison with the input (the addressing mechanism described above is performed);
- per-bit data writing into the cell;
- logical functions, which determine the actual value which is to be written into the cell;
- interaction with neighboring PU (priority scheme);
- reading the data.

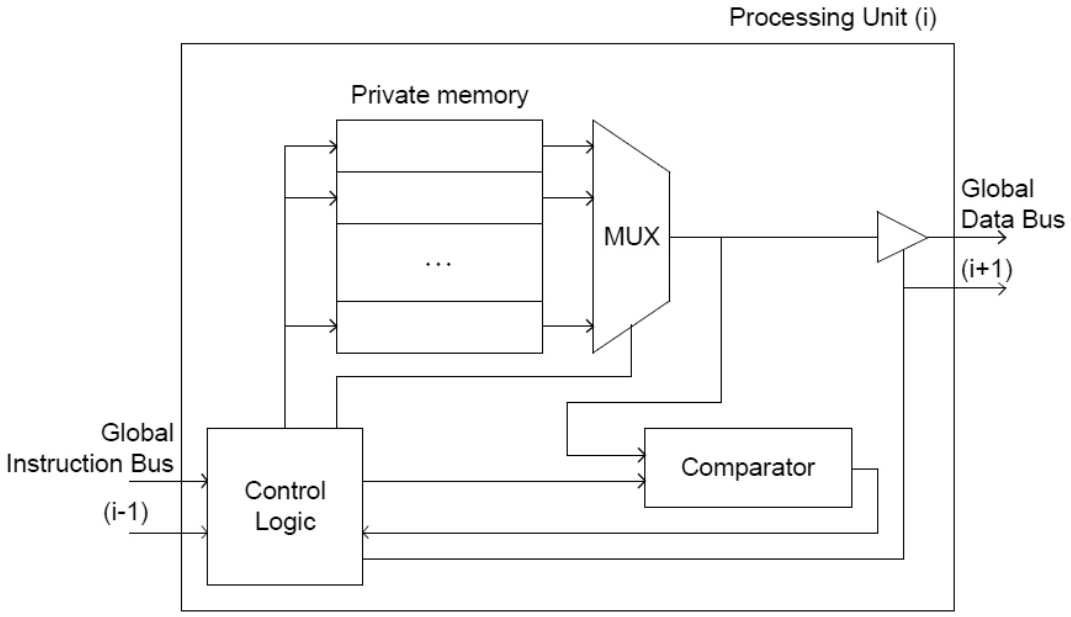

Figure 3: Processing unit block diagram

3 Some regularities and objective limitations

Key idea in the process of the processor developing is to keep the functional complexity of PU as simple as possible, and as a result, it will allow to represent a huge number of these elements in one single chip. At the same time, the processor must be of general purpose to some extent (for providing the profitability of the project, as it was explained earlier), and that naturally affects the structure of PU. Moreover, the requirements imposed by a particular application program, for which processor is being manufactured, can vary greatly from one system to another. Some systems require maximum performance, for others, the maximum of the space of the stored memory is primarily important, for the third the cost will be the main limiting factor.

To solve this problem the processor is to support specific settings for a specific application system (application task), for which this processor is being manufactured. For the architecture proposed, such a setting may be performed at several conceptual levels. In particular, you can select the hardware configuration of the processor (selection of processor data bit-width, the amount of PU, the local memory space) and software configuration of the processor (a different interpretation of the processor data format). Let's stop on the hardware configuration of the processor, and go trough some regularities and limitations of the architecture proposed.

1. Processor bit-width b - fully depends on the solved in a system problem. For example, to keep the semantic network in the processor memory, memory cell must be able to accommodate the representation of any custom network element. To represent the graph vertex it is necessary to keep a certain unique identifier of the vertex and the set of its properties (attributes) values. To represent the arc of the graph - it is

necessary to keep identifiers of two, which are incident to the arc, of vertices and to some properties of the arc. Increased of processor bit-width b leads to:

- volume gain of PU local memory and then the extension of the PU memory circuit;
- the gain of combinational circuit of PU control logic;
- reduction of execution speed of PU control logic (additional delays on multiplexors, etc.);
- the gain of the instruction size (4 times faster than the processor bit-width b, according to rough estimate);
- increase in the amount of data buses, and at the same time, increase of a total complicity of circuit trace;
- increase in the cost of a single memory cell.

The cost of one memory cell of the processor can be taken as the cost of the chip made divided by the total amount of memory cells, which carry the core value. In fact, the cost change (increase or falling) is determined by the ratio of the area of the combinational circuit (CC) of the processor (the whole control logic) to the area occupied by its own memory elements (ME) of the processor:

$$K = \frac{S_{CC}}{S_{ME}}$$

Increase of the value of the variable K leads to the growth of the cost of one memory cell, reduction of the value of it causes the impact of value.

2. The number of private memory elements M, per one PU. The increase of the value of the variable M leads to:

- an increase in the local memory space of PU and, so the increase of the memory circuit of PU;
- the increase of combinational circuit of PU control logic;
- the increase of the period of time spent by PU to perform the operation (the more memory cells the PU includes, the more time the serial processing of it requires);
- the reduction of performance (the number of operations performed per one unit of time);
- decrease in the cost of a single memory cell.

Obviously, to provide maximum system efficiency it is necessary to to reduce the value of the variable M as much as possible, i.e.to reduce the number of memory cells for PE, and thus to increase the total number of PE on the chip. However, it will be accompanied by the rising one memory cell cost, which is not always valid

3. In most cases, the total amount of PE on the chip N is the derivative of the processor bit-width b word value, local memory size M and the maximum allowed area of the circuit S_{MAX}, i.e. after selection of the particular value of the basic parameters of the system, we place the maximum possible number of corresponding PU on the chip:

$$S = S_{CC} + S_{ME} \rightarrow S_{MAX}$$

In general case the increase of the amount of PU leads to:

- the increase of the total processor memory space;
- the increase of the processor efficiency (the bigger memory space is processed at the same period of time);
- the increase of the amount of data buses, and at the same time, increase of a total complicity of circuit trace.

It should be noted that the maximum number of PU is limited by certain maximum number N_{MAX}, it is determined by the presence of local links between PU (serial circuit delays). The solution of this problem is not considered in the paper.

4 Pathfinding problem on semantic processor

In the work [9] on the base of the programming model of processor architecture proposed the problem of finding the shortest path in the graph was solved. To solve the problem the parallel algorithm was used, it was developed on the bases of Dijkstra's algorithm and wave algorithm.

4.1 Problem definition and solution

Given a weighted graph $G(V, A)$ without loops and arcs of negative weight. The problem is to find the shortest path from a vertex of a graph G to all other vertices of the graph. Let introduce the following notations:

- V - set of graph vertices;
- A - set of graph arcs;
- $c[ij]$ - cost (weight, length) of arc ij;
- a - start vertex;
- U - set of processed graph vertices;
- W - set of graph vertices which represents the wave front;
- $d[u]$ - at the end of algorithm it is equal to the distance of the shortest path from vertex a to vertex u;

- $p[u]$ - at the end of algorithm it contains the shortest path from vertex a to vertex u.

As the result pseudocode of the proposed algorithm can be written as the following:

Algorithm 1 Pathfinding algorithm pseudocode

```
1: for ∀v ∈ V, v ≠ a do
2:     d[v] ← ∞, p[v] ← ∅
3: end for
4: d[a] ← 0, p[a] ← a
5: U ← a, W ← a
6: while ∃u ∈ W do
7:     W ← W \ {u}
8:     for ∀v ∈ V, uv ∈ A do
9:         if v ∉ U or d[v] < d[u] + c[uv] then
10:            U ← U, v
11:            W ← W, v
12:            d[v] ← d[u] + c[uv]
13:            p[v] ← p[u], v
14:        end if
15:    end for
16: end while
```

4.2 Implementation on semantic processor

Let's briefly examine the example of program 'customization' of the processor, the topical point of which is to build an additional level of abstraction over the system of processor instructions, that [level] tightly focused on solving the definite problem. To achieve it, it is necessary to determine its own data format (definite bits of memory cells are provided with the necessary sense) and define new instruction set, which works in terms of the problem to solve. The performance of such an extended instruction set can be implemented by an application programmer at one's development level or by a system programmer at CU processor level, which can be represented as a reprogrammable controller.

Accordingly to the proposed parallel algorithm the data format was defined (see Figure 4) and there is an extended list of commands below:

- `init` - the initial value for all the memory cells in accordance with the algorithm used is set;
- `createVertex` - to create a vertex with the given identifier;
- `createArc` - to create an arc with the given identifier and the cost of the transition;
- `setMinDistance` - to set for a given vamdahlertex the minimum distance found and identifier of the previous vertex (to be able to restore the path after);

Used	Cell type	ID	Vertex attributes				
			v	n	f	$distance_{min}$	prevID
	0	k-1 0				m-1 0	k-1 0

Used	Cell type	ID1	ID2	Arc attributes	
				c	cost
	1	k-1 0	k-1 0		m-1 0

Figure 4: Data format of PU when solving the pathfinding problem in the graph

- `setWaveStartVertex` - to set the flag of entry in the wave front for a given vertex;
- `readVertex` - to read the vertex by its ID;
- `readNextVertexFromWavefront` - to read the next vertex in the wave front;
- `moveWavefront` - to start the next generation of wave front;
- `findAllOutputArcs` - to find all the arcs going from a given node;
- `readNextOutputArc` - to read the next arc by the previously found.

5 Conclusion

The processor architecture, designed to solve graphs' problems, and in particular, the efficient performance of semantic data processing algorithms, was presented in the paper. There were examined the basic principles of operation of the processor, the data format and instruction set, the structure of the processing unit, some regularities and objective limitations of implementing the processors of such an architecture. In particular, it describes the most important parameters of the processor architecture, the choice of values of which is actually a setting of an architecture for solving a particular application problem.

As part of the development of the architecture proposed, the next stage of our research will be the implementation of the architecture on the existing parallel platforms, such as GPU (CUDA technology), cluster and FPGA for estimation of real gains in performance and efficiency while solving the different routine problems from the field of semantic analysis in comparison with solutions actively used today.

References

[1] Golenkov V.V. and Guliakina N.A., *Graphodynamical models of parallel knowledge processing*, Proc. Int'l Conf. Open Semantic Technologies for Intelligent Systems (OSTIS'2012), - February, 2012, BSUIR, Minsk, (2012), p.23–52.

[2] Gene M. Amdahl, *Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities*, AFIPS Conference Proceedings, Vol.30(1967), p.483–485.

[3] John L. Gustafson,*Reevaluating Amdahl's Law*,Communications of the ACM,Vol.31, Iss.5(1988), p.532–533.

[4] W. D. Hillis,*The Connection Machine*,The MIT Press,Cambridge (1989).

[5] Hiroaki Kitano and Dan Moldovan,*Semantic Network Array Processor as a massively parallel computing platform for high performance and large-scale natural language processing*, Proc. Int'l Conf. on Computational Linguistics (COLING '92),Vol.2,(1992), p.813–819.

[6] André R. Brodtkorb and Trond R. Hagen and Martin L. Sætra, *Graphics processing unit (GPU) programming strategies and trends in GPU computing*, Journal of Parallel and Distributed Computing, Vol.73, Iss.1(2013), p.4–13.

[7] Bairak S. and Adzinets D. and Tatur M. and Philipoff P. and Munoz M., *Parallel processors for intelligent systems development,* Proc. Int'l Conf. Open Semantic Technologies for Intelligent Systems (OSTIS'2012) - February, 2012, BSUIR, Minsk (2012), p. 135–140.

[8] Verenik N. and Seitkulov Y. and Tatur M., *Development of ASIP for semantic information processing*, Electronics info, Vol.8(2012), p.95–98.

[9] Tatur M. and Seitkulov Y. and Verenik N. and Girel A.,*Pathfinding on a Specialized Vector Processor*, ProcProc. Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA '2013), Vol.1,(2013), p.711–716.

Nick L. Verenik, Alexey I. Girel,
Department of Computer Sciences ,
Belorussian State University of Informatics and Radioelectronics,
Minsk , Belarus,
220013, Brovka Str.,6.
Email: `nick.verenik@gmail.com, alexey.girel@gmail.com;`
Yerzhan N. Seitkulov,
Faculty of Information Technology,
L.N.Gumilyov Eurasian National University,
Astana, Republic of Kazakhstan,
010008, Mirzoyana Str., 2.
Email: `erj@mail.ru`;
Mikhail M.Tatur,
Department of Computer Sciences,
Belorussian State University of Informatics and Radioelectronics,
Minsk , Belarus,
220013, Brovka Str.,6.
Email: `tatur@bsuir.by` Received 25 Jan 2014, in final form 29 Mar 2014, accepted 5 Sep 2014