International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 03 – Issue 05, September 2014

Architecture of Multi-Core System-on-the-Chip with Data Flow Computation Control

Branislav Madoš
Department of Computers and Informatics,
Faculty of Electrical Engineering and Informatics,
Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic
Email: branislav.mados {at} tuke.sk

Abstract—**System-on-the-chip can be defined as the integrated circuit (chip) which integrates all necessary components of the computer or other system. This paper deals with the architecture of multi-core system-on-the-chip that is based on the data flow paradigm of program execution control. The flow of the program is controlled by the flow of data instead of the flow of instructions. This is an important alternative to the currently mainstream control flow paradigm, today mostly represented by architectures that are constructed on the Von Neumann principles. Proposed architecture uses tiles as the modern approach to the design of multi-core microprocessors. This concept brings possibility of high scalability of the design, without the need of fundamental redesign of the chip. Architecture brings layout of components in bi-dimensional mesh that consists of processing elements including memory, input-output elements and communication networks. The first part of the paper introduces basic features of the architecture and briefly describes elements that architecture consists of. Second part of the paper is dealing with unique construction of memory, data flow program mapping techniques and subgraph coloring technique, which are the results of the research at the Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice.**

Keywords- data flow, mutli-core, system-on-the-chip, SoC, data flow graph, subgraph coloring

I. INTRODUCTION

In the previous decades, starting from the middle of the last century, we have witnessed tremendous expansion of computer and electronics industry. This is associated with long-term unprecedented increase in the communication and computing performance of integrated circuits accompanied with the increase of memory chip capacities. This trend was driven by the increase of the number of basic building blocks integrated onto the silicon chips and by the increase of the density of their integration. This situation has brought to the industry the opportunity for continual rise of the communication and computing performance of superscalar monolithic uniprocessors. Unfortunately, it is not easy to recast this opportunity into the appropriate increase in computing performance of monolithic uniprocessors in accordance to the increase of the number of integrated transistors of the silicon chip. Designs of microprocessors are reflecting outlined problem with multi-core approach or single chip multiprocessors [1]. Commercially available microprocessors are integrating more than one complex superscalar core on the single chip. Other strategies are promoting downsizing of the superscalar core in the form of reduction of its complexity and footprint in favor of the possibility to integrate even more cores on the chip. The ratio of the complexity of the core and the number of cores integrated into the chip can therefore become an important parameter of multi-core microprocessor architectures. Integration of not only tens but hundreds and even thousands of cores will be considered in the future. Integration of memory onto the single chip with processing elements can become another step in order to avoid bottleneck problems connected with the access to the operating memory.

Different layouts of cores, memory elements and especially interconnection networks are considered together with different approaches to the program execution organization and control. Not only control flow driven architectures also with Very Long Instruction Word (VLIW) are used but perspective of the architectures with data driven computation model, called data flow architectures, is continually investigated.

Very important approach to the design of microprocessor architectures can be defined as the tile computing and was introduced in [2]. It is possible to describe the tile computing as the principle of the use of standardized components including processing elements (small and simple processing elements, arithmetic and logic units or more complex cores), memory elements and modules and various types of communication networks, in easily and highly scalable architectures of microprocessors and systems-on-the-chip. The advantage is that up scaling of the chip needs minimal change of used components and overall architecture. Tens of cores are integrated not only in experimental architectures but also in commercially available microprocessors. In some cases the number of cores exceeds hundred.

Commercially available general purpose 64-bit tile computing microprocessors are designed by Tilera Corporation. TilePro64 integrates in 1517 BGA package 64 cores with L1 and L2 cache integrated in each core in two-dimensional layout of cores in 8 × 8 mesh. Chip performance is 443 Billions Operations Per Second (BOPS) at 700 MHz. It is able to run independent operating system on each tile or it is possible to run multiprocessing operating system on multiple tiles. TileGX-8072 processor integrates 72 cores in bi-directional mesh with layout of 8 × 9 cores [3][4].

TeraScale microprocessor is 96-bit VLIW processor designed by Intel Corporation as the result of the research in the Tera-Scale Computing Research Program. Microprocessor integrates 80 cores in bi-directional layout of 8 × 10 cores. Processor is fabricated with 65 nm process. Microprocessor integrates 100 million transistors on 275 mm2 die. With ability to perform 4 single precision operations in floating-point per cycle and core, microprocessor delivers theoretical peak performance of 1.37 TFLOPS at 4.27 GHz of operating frequency. Instructions set architecture consists of 16 instructions. Each core is able to run own program and network-on-the-chip is transferring data and coordination information between cores with use of message-passing. Research program and microprocessor are closely in [5][6].

Digital signal microprocessors (DSP) are designed under the tile computing principles. With data flow paradigm of program execution control and 8 × 8 layout of 64 cores, VGI is representative of this kind of microprocessors. Each core comprises approximately 30.104 transistors. Another architectures, namely TRIPS [7], SmartMemories [8][9], nanoFabrics, WaveScalar [10], Monarch [11] and others can be mentioned. The trend is heading toward not only tens, but hundreds and even thousands of cores integrated into the chip, often with integrated operating memory, which can be considered as the very promising approach that can solve the bottleneck problem of data transfer between memory and microprocessor.

The structure of the article is as follows:

In the section II of the paper proposed architecture is described, with the brief introduction of elements (including processing elements and input-output elements) and communication networks (local and global communication network). Special attention is than paid to the data flow graph mapping and three unique data flow graph mapping techniques (sequential, global and mask mapping) are described. Data flow subgraphs coloring technique is introduced in the following section. In the last part of the paper operating memory of the processing element is described and memory operations are briefly introduced.

II. PROPOSED ARCHITECTURE

Computer architecture of the dataflow computer, that is proposed as the part of the research is representing 32-bit system-on-the chip. Program execution is driven by the flood of operands, according to the data flow paradigm of computing. Architecture is designed with the use of tile computing principles, with aim to prepare highly scalable design. It can be scaled up in term of integration of more processing elements and other components of the architecture on the chip with minimum design changes.

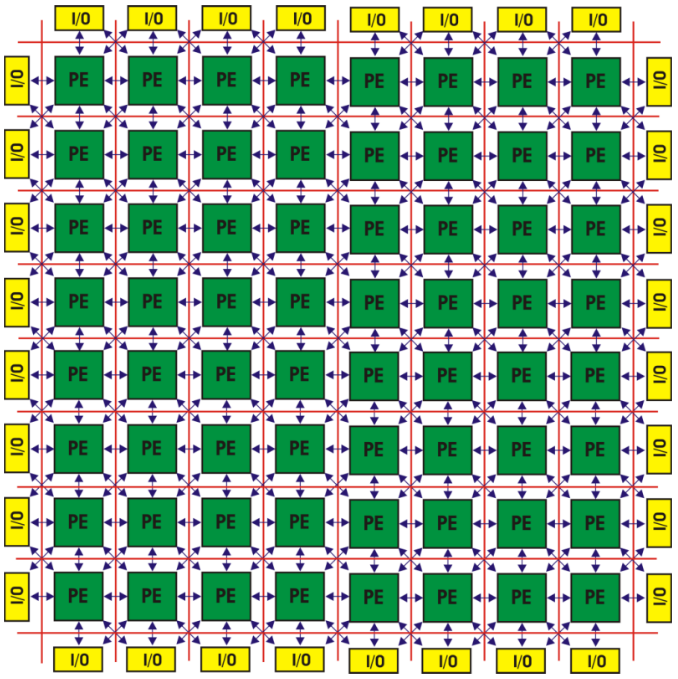

Figure 1. Processing array contains 64 processing elements located in 8 × 8 two-dimensional layout, 32 input-output elements, communication channels intended for short range communication (SCN) and the bus intended for global communication (GCN).

It is possible to scale up this architecture not only by redesign of the chip but also by the use of proposed system-on-the-chip in multichip configuration. In this case chips are placed in bi-dimensional mesh layout and are forming consistent array of processing elements.

A. *Elements of the Architecture*

Proposed computer architecture utilizes small elements with simple design with aim to allow integration of maximum number of elements as possible in accordance to the number of transistors available on the chip. Basic element of the architecture is Processing Element (PE) that is executing data flow program. In each processing element operating memory is integrated, used as the activation frames store (AFS). All PEs placed in bi-directional mesh are forming Processing Array (PA). Other components integrated on the chip are Input-Output Elements (I/O) that are interconnecting PEs with neighboring components of computer (Figure 1).

In prototype, which is built with use of Xilinx Field Programmable Gate Array (FPGA) technology, 64 processing elements (PE) are accompanied with the 32 input-output elements (IO). It is possible to determine ratio (R) between the number of processing elements and input-output elements integrated, with use of the formula (1). The ratio is increasing according to the value of n.

$$R = \frac{n^2}{4n} \qquad (1)$$

where

n is the number of PEs forming the edge of PA

1) *Processing Element*

Each processing element has simple design and consists of:

- Activation Frame Store (AFS) which represents unit consisting of the operating memory that is developed as the part of this research. AFS has 1024 × 144 bits organization. It is not only used as the instructions store but it is also performing Fetch phase of instructions cycle and it is capable of other operations that are described in the section 3 of this paper. The concept of the cache memory is not implemented and there is no buffer for prefetech of activation frames from activation frames store.
- Arithmetic and Logic Unit (ALU) which represents unit that is performing Execute phase of instructions cycle of integer arithmetic and logic operations.
- Control Unit (CU) which represents unit that controls execution of instructions and communication with neighboring elements.

Each activation frame stored in AFS is addressable and all activation frames of all PEs integrated on the chip are forming virtual cube of addressable space of computer. Addresses consists of three components X(2:0), Y(2:0) and Z(9:0). Components X and Y are forming address of memory integrated in processing element of processing array and Z component is representing address of activation frame in AFS of respective PE. Each PE is allowed to address only activation frames stored in operating memory integrated in PE and it is not allowed to address activation frames of other PEs.

2) *Input-Output Elements*

Processing elements, which are placed on the edges of the array of processing elements, are connected to the input-output elements, which are surrounding this processing array. Each input-output element allows communication with other components of computer system. Input-Output Element is serving as the input point of operands that are produced outside the processing array, or as the output point, when operands are leaving processing array and are used as the input of other components of the computer system. In multichip configuration, two input-output elements, each on different chip, are forming IO bridge. It interconnects two processing elements of different chips and allows transmission of operands between them. IO bridge is forming interconnection with the same function as the local network communication channel (Figure 2).

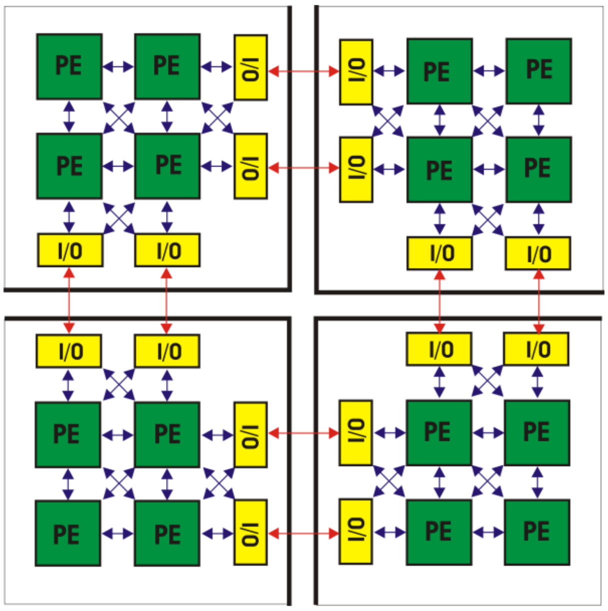

Figure 2. Multichip computer configuration allows scaling of the computer beyond borders of the chip. Figure describes four chips in bi-directional layout that are interconnected by I/O elements.

B. *Communication Networks*

Crucial for effective design of computer architecture that is developed under the tile computing principles is also possibility of effective communication of computer components. In proposed architecture two types of communication networks are created, for short range and global range communication.

Short range communication (SCN) is allowed through the communication channels which are interconnecting each PE with each of eight neighboring elements. It is allowed that pairs of elements of the chip are communicating in parallel and it allows high bandwidth short range parallel communication of elements of the chip. Each PE is allowed to communicate with one of neighboring elements via token passing. Token is formed by 32-bit operand, accompanied with 10-bit target address of activation frame and 1-bit indicator of target operand (A/B). Short range communication is used in *Compute Mode*, when data flow program is executed.

Global range communication (GCN) is realized in the form of the bus that is interconnecting all PEs and it allows data flow graph mapping into the operating memory of each PE in three different manners, which are developed as the part of this research and are described below. This communication network is used in the *Load Mode*, when data flow graph is mapped into operating memory of each processing element. It is not possible to use this communication network in *Compute mode*. On the Figure 1 it is shown schematically as the red mesh. GCN interconnects all PEs and it is connected to the pins of the chip.

C. Data Flow Graph Mapping

Unique characteristic of proposed architecture is that it is possible to map data flow program into the activation frames stores of processing elements by three different approaches. Approach of the program mapping can be dynamically changed immediately after each activation frame is mapped.

First mode called sequential mapping allows mapping of instructions into activation frames in classical manner, with one activation frame at once. There is no possibility of concurrent instructions mapping to the different AFS of different PEs. All three components (X, Y and Z) of the activation frame address are used.

Second mode called global multi-mapping allows mapping of the same instruction into all AFS of all PEs in the PA concurrently in one machine cycle. Activation frame is addressed with use of Z component of the address only; X and Y components are omitted.

Third mode called mask multi-mapping mode allows concurrent mapping of the same instruction into AFS of selected PEs of PA. Selection of PEs is allowed by the use of the mask. For specification which activation frames are targeted Z component of address is used along with two 8–bits vectors representing the mask for X and for Y axis of the processing array.

It is possible to switch between modes of the data flow graph mapping in time of this process, with aim to optimize the time of the data flow graph mapping. Data flow graph mapping is time consuming process and the time of mapping will increase with increasing number of processing elements integrated into the processing array. That is why attention must be paid to this process in future research of multi-core processors.

1) Sequential Data Flow Graph Mapping

In sequential mapping it is possible to map dataflow graph into activation frames only sequentially, with use of X, Y and Z axis address in address space of activation frame stores (Figure 3).

T_{PAmap} is total time in machine cycles (MC) for mapping of data flow graph onto processing elements of processing array.

T_{PEmap} is total time in MC for mapping of subgraph of data flow graph onto respective processing elements of processing array.

T_{PAmap} can be expressed by

$$T_{PAmap} = \sum_{n=0}^{X-1} T_{PEmap}\; n \qquad (2)$$

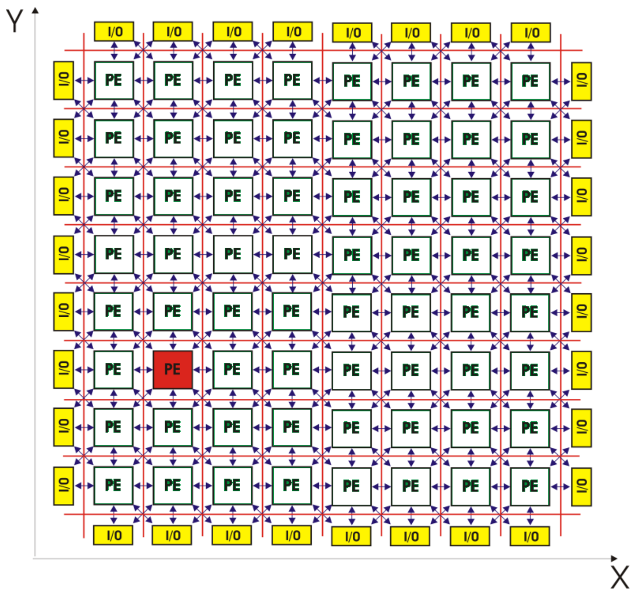

Figure 3. Seqential mapping of data flow graph.

Where X is the number of processing elements onto which DFG is mapped.

T_{PEmap} can be expressed by

$$T_{PEmap} = N \times 1 \text{ MC [MC]} \qquad (3)$$

Where N is the number of activation frames which are mapped into the activation frames store of respective processing element.

Maximal total time of data flow graph mapping *Max* T_{PAmap} into processing array in MC can be expressed by

$$Max\ T_{PAmap} = X \times Max\ T_{PEmap} \text{ [MC]} \qquad (4)$$

Where X is the number of processing elements which are forming processing array into which data flow graph is mapped.

Max T_{PEmap} can be expressed by

$$T_{PEmap} = C_{PAR} \times 1 \text{ MC [MC]} \qquad (5)$$

Where C_{PAR} is capacity of the activation frames store of processing element, and is expressed in number of activation frames.

2) Global Data Flow Graph Mapping

Global data flow graph multi-mapping allows mapping of activation frame into respective address in Z axis in all AFS of processing elements of PA concurrently in one machine cycle (Figure 4).

T_{PAmap} can be expressed by

$$T_{PAmap} = T_{PEmap} = N \times 1 \text{ MC [MC]} \qquad (6)$$

Where N is the number of activation frames which are mapped in activation frames store of each processing element of the chip.

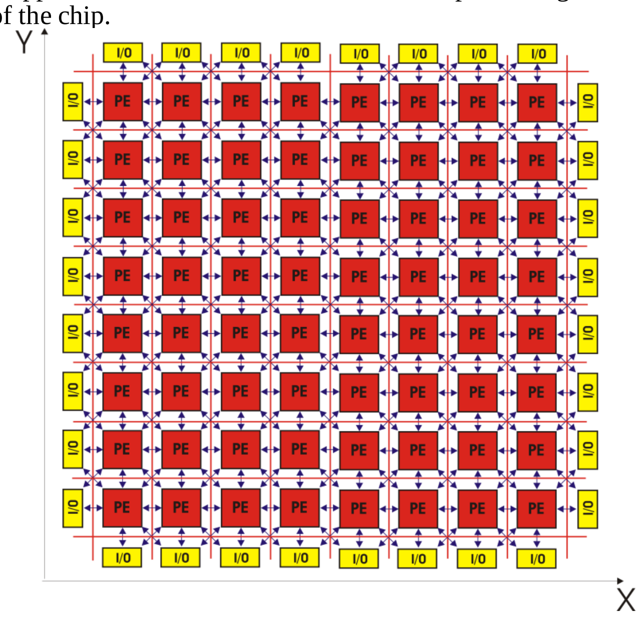

Figure 4. Global data flow graph multi-mapping.

Maximal total time of data flow graph mapping into processing array and processing element which are *Max* T_{PAmax} and *Max* T_{PEmap} can be expressed by

$$Max\ T_{PAmap} = Max\ T_{PEmap} = C_{PAR} \times 1 \text{ MC [MC]} \qquad (7)$$

Where C_{PAR} *is* capacity of activation frames store of processing element, that is expressed in number of activation frames.

3) *Mask Data Flow Graph Mapping*

Mask data flow graph mapping allows mapping of activation frame into respective address in activation frames store of all processing elements that are addressed by mask of processing array concurrently in one machine cycle (Figure 5).

Processing elements which are activated for mask multi-mapping are defined by mask.

T_{PEmap} can be expressed by

$$T_{PEmap} = N \times 1 \text{ MC [MC]} \qquad (8)$$

Where N is the number of activation frames which are mapped into activation frames store of processing element.

It is possible to define multi-mapping mask MASKX(m-1:0) for X axis of processing element and MASKY(m-1:0) for Y axis of processing element. m is the dimension of bi-directional mesh m × m of processing elements in processing array.

PE(x:y) is activated for mapping in case of MASKX(x) = 1 and MASKY(y) = 1.

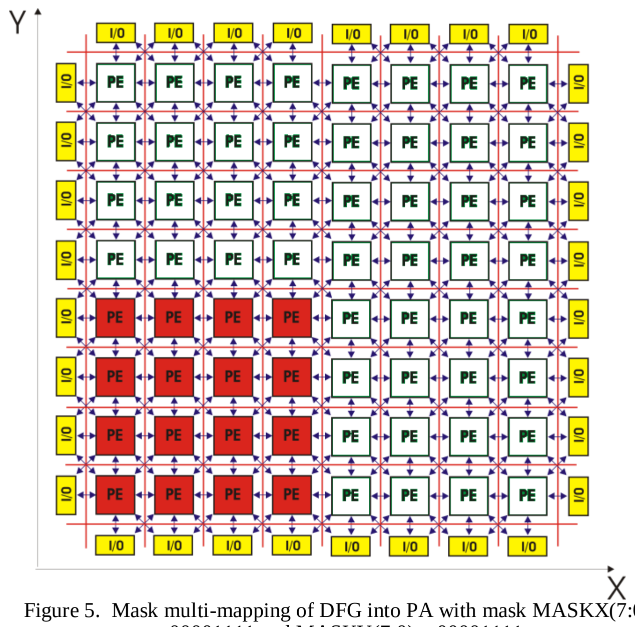

Figure 5. Mask multi-mapping of DFG into PA with mask MASKX(7:0) = 00001111 and MASKY(7:0) = 00001111

D. *Data Flow Subgraph Coloring*

Program of the data flow computer is prepared in the form of directed graph. It is possible to partition the data flow graph into the subgraphs and this process is iterative. It is possible to denote subgraphs with their respective colors. In proposed architecture, each subgraph is denoted with 32-bit COLOR tag, which is part of instruction vector (bits 127:96).

Each instruction stored in the memory is tagged with color of the subgraph as it is shown in the example at Figure 6. “yellow” subgraph has COLOR tag $(00000000\ 00000000\ 00000000\ 00000100)_2$ (leading zeros of 32-bit COLOR tag vector are omitted in the Figure 6 in favor of more readable notation), “green” subgraph has COLOR tag $(00000000\ 00000000\ 00000000\ 00000110)_2$ and “white” subgraph has COLOR tag $(00000000\ 00000000\ 00000000\ 00000111)_2$.

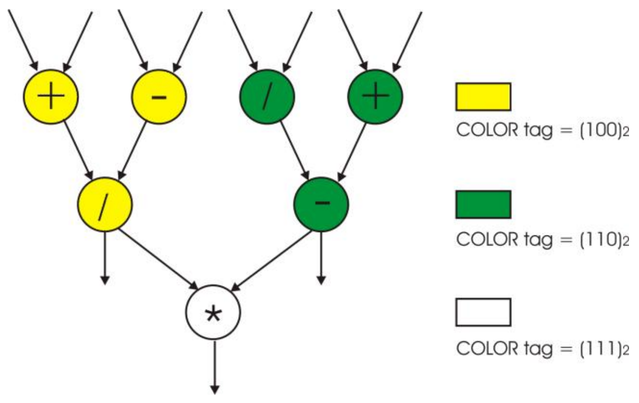

Figure 6 Example of data flow graph partitioned into 3 subgraphs and tagged with different COLOR tags

Proposed architecture brings concept of de/initialization operations for the entire data flow graph via IG (Initialize Graph) and KG (Kill Graph) instructions that are part of the instructions set architecture. Another possibility is de/initialization of subgraph via ISG (Initialize SubGraph), ISGO (Initialize SubGraph with Operands) and KSG (Kill SubGraph) instructions.

In the basic concept of the COLOR tag usage, all instructions mentioned above have operand, which is 32-bit long and represents COLOR tag of subgraph on which instruction is applied. It is possible for example to initialize each of subgraphs with use of instruction ISG $(00000000\ 00000000\ 00000000\ 00000100)_2$, ISG $(00000000\ 00000000\ 00000000\ 00000110)_2$ or ISG $(00000000\ 00000000\ 00000000\ 00000111)_2$. Because of specific construction of operating memory of the computer, operation is applied in parallel on all activation frames stored in memory that are tagged with respective COLOR tag in one machine cycle. However, this straightforward approach has its shortcomings. If we are considering "green" and "yellow" subgraphs as two partitions of one graph and both are needed to be initialized, it is not possible to perform this operation in other way but to use sequentially two ISG instructions with $(00000000\ 00000000\ 00000000\ 00000100)_2$ and $(00000000\ 00000000\ 00000000\ 00000110)_2$ operands respectively. This means not only use of more instructions of the program, but also more machine cycles are needed. That is why more complex approach was chosen.

Instructions which are using COLOR tag are not unary, but have two operands. First is COLOR tag which is 32-bit vector, as mentioned above, second operand is 7-bit vector that represents the number of bits of COLOR tag, which are compared. It is applied when it is evaluated, if instruction, on which de/initialization operation is applied, is part of targeted subgraph. Bits are counted from MSB of COLOR tag vector. It is meaningless to compare 0 bits of COLOR tag vector because in that case operation is applied on all activation frames stored in memory. This is performed by IG and KG operations respectively. There is no possibility to compare more than 32 bits of COLOR tag vector. That is why value of the second operand is restricted to the range of <1:32> in decimal.

Version of instructions with two operands is powerful, because it can target more than one subgraph at once and that means operation of de/initialization can be performed on more subgraphs at the same time, in one machine cycle. In the example mentioned above, if ISG operation with operands $(00000000\ 00000000\ 00000000\ 00000110)_2$, $(0111111)_2$ or $(00000000\ 00000000\ 00000000\ 00000111)_2$, $(0111111)_2$ is applied, subgraphs with 31 MSB bits set to the value $(00000000\ 00000000\ 00000000\ 0000011)_2$ are targeted. Because only one bit left, there are two possible subgraphs, which can be targeted, with LSB bit set to 0 and 1 respectively. In the example on Figure 6 those subgraphs are „green" and „white".

If the first operand of ISG instruction is set to the value $(00000000\ 00000000\ 00000000\ 000001xx)_2$ where xx is the 2 bit vector of any value and second operand is set to the value $(0111110)_2$, there are 30 MSB bits compared and subvector consisting of 2 LSB bits is left uncompared. It means 4 subgraphs with 2 LSB bits set to the values $(00)_2$, $(01)_2$, $(10)_2$, $(11)_2$ are targeted. In the example shown in Figure 6, all 3 subgraphs meet the condition and are targeted with this instruction.

In the example shown on Figure 6, there is no possibility to target "yellow" and "white" subgraphs at once without targeting "green" subgraph.

E. Memory Subsystem of the Processing Element

Conventional concept of control flow model utilizes Random Access Memory (RAM) as the operating memory. It has advantages in simple design of the memory module, high density of memory elements integration and low power consumption of memory chips. On the other hand, it appears to be less useful for data flow computing. One of basic operations of data flow computer is searching for immediately executable instruction which is stored in memory and it means that in worst case all memory must be searched sequentially. Operation requires logic outside the memory and can be extremely time consuming.

Solution of this problem can be found in Content Addressable Memory (CAM) which is used typically in active network devices for example for searching in routing tables [12] or in cache memory [13]. CAM is designed with aim to allow search of entire memory in a single operation and it allows much faster search for immediately executable instruction in comparison with RAM. Unlike RAM with simple storage cells, in CAM each memory bit is accompanied with comparison logic circuit for detection of match between stored bit and searched bit and additional circuit for evaluation if all bits in data word are matching. This additional circuitry increases physical size of the CAM chip, or in other words is lowering density of memory elements, increases manufacturing costs and power consumption of the memory [14]. With Ternary Content Addressable Memory (TCAM) it is possible to use not only 1 and 0 but also X as the "Don't care" for one or more bits in data word to make search more flexible.

In our research proposed memory is philosophically based on TCAM. Memory incorporates not only memory cells, but also combinational logic to perform search operation, in which immediately executable operation can be found, with use of parallel searching in whole memory. Altered CAM concept incorporates generator of priority into the memory for selection of matching data word with lowermost address in memory. In each time only one data word representing activation frame that contains immediately executable instruction can be found on the output of memory. It is possible to write to the memory with use of address, when specific part of data word can be stored into the specified address. It is used for storing of operands in time of program execution.

Another important feature of proposed computer architecture is the ability to initialize or deinitialize subgraphs of data flow graph with or without operands preservation. It is needed to perform operation of evaluation, if activation frame is the part of subgraph and that is why this operation must be performed on each activation frame stored in memory. After evaluation those instructions which are part of the target subgraph are initialized or deinitialized according to the type of instruction. All of those operations are performed on each activation frame stored in activation frames store in parallel in one machine cycle with built in combinational logic circuit.

Memory operations of proposed memory can be divided into two main groups according to the mode in which memory and computer are operating. There is *Load mode* (L) and *Compute mode* (C) proposed. Operations performed in traditional manner, including reading and writing of the specific activation frame with use of address, are forming first group of operations (I), which are performed in the *Load mode*, when memory behaves as conventional RAM. Another group of operations is formed by operations that are data flow specific (II), and are performed on particular activation frames stored in memory (IIa) with use of the address or specific attribute of activation frame; or are performed on all activation frames stored in memory in parallel (IIb). Two mentioned subgroups of operations are available only in the *Compute mode* and are performed in connection with particular instructions of data flow graph in the time of data flow graph execution.

I. Load mode

1. Read / Write

II. Compute mode (data flow specific operations)

a. Operations on the single activation frame

1. Write of A operand
2. Write of B operand
3. Read of activation frame

b. Operations on all activation frames

1. Initialization of data flow graph
2. Initialization of data flow subgraph with operands
3. Initialization of data flow subgraph without operands
4. Deinitialization of data flow graph
5. Deinitialization of data flow subgraph

Instead of having separate Load and Compute inputs to control the two groups of operations, the $L/\overline{C}$ control signal was defined. If the $L/\overline{C}$ input is set, memory is in *Load mode*, otherwise in *Compute mode*. In *Load mode* the signal $R/\overline{W}$ determines the particular operation of reading and writing respectively.

CONCLUSION

Presented architecture of data flow computer with the tile organization of processing elements is the result of the research at the Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Kosice. As the part of the research we proposed memory with function of activation frames store that is based on Ternary Content Addressable Memory (TCAM) which behaves as the conventional RAM in Load Mode, and CAM in Compute Mode, and is able of operations execution that are specific for proposed data flow computer architecture. Memory represents innovative approach to the construction of data flow computer and in the future research can become the basis for the solutions of disadvantages of data flow computers that are limiting their use in practice. Another unique technique proposed within this research are data flow subgraph coloring and three different modes of data flow graph mapping, sequential data flow graph mapping, global data flow graph mapping and mask data flow graph mapping.

In the future research cache memory will be implemented to the design of the chip. Architecture of cache memory will have design proposed in this article for operating memory, and future operating memory will have simpler and cheaper construction.

Attention will be paid to the investigation of the possibility to integrate execution of control flow and data flow program code in single processor. Data flow processor will emulate control flow processor when necessary. The aim is to prepare possibility to use advantages of both ways of program execution control in single processor.

ACKNOWLEDGMENT

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0008-10 and project KEGA 008TUKE-4/2013: Microlearning environment for education of information security specialists.

REFERENCES

[1] B. Grot, J. Hestness, S. W. Keckler, O. Mutlu: „Express Cube Technologies for On-Chip Interconnects“, Proceedings of the 15th International Symposium on High-Performance Computer Architecture, February 14 – 18, 2009, Raleigh, North Carolina.

[2] E. Waingold et al.: “Baring it all to software: RAW machines”, IEEE Computer, Sep. 1997. Technical Report, MIT Cambridge, Massachusetts, USA.

[3] Tilera (2012) Manycore without boundaries: TILEPro64 processor. Available online: www.tilera.com/products/processors/TILEPRO64.

[4] Xuan-Yi Lin, Kuan-Chou Lai, Kuan-Ching Li, and Yeh-Ching Chung. 2013. Efficient programming paradigm for video streaming processing on TILE64 platform. *J. Supercomput.* 65, 2 (August 2013), 823-847. DOI=10.1007/s11227-012-0867-6

[5] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar: "An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS," IEEE Journal of Solid-State Circuits, Vol. 43, No. 1, Jan 2008.

[6] A. Munir, F. Koushanfar, A. Gordon-Ross, S. Ranka: High-performance optimizations on tiled many-core embedded systems: a matrix multiplication case study, The Journal of Supercomputing, v.66 n.1, p.431-487, October 2013.

[7] M. Gebhart, B. A. Maher, K. E. Coons, J. Diamond, P. Gratz , M. Marino, N. Ranganathan, B. Robatmili, A. Smith, J. Burrill, S. W. Keckler, D. Burger , K. S. McKinley: „An evaluation of the TRIPS computer system", ACM SIGPLAN Notices, v.44 n.3, March 2009 [doi>10.1145/1508284.1508246]

[8] K. Mai, T. Paaske, J. Nuwan, R. Ho, W. Dally, M. Horowitz: "Smart Memories: A Modular Reconfigurable Architecture", ISCA 00, Vancouver, British Columbia, Canada, ACM 1-58113-287-5/00/06-161

[9] N. Wu , Q. Yang , M. Wen , Y. He , J. Ren , M. Guan , Ch. Zhang, "Tiled multi-core stream architecture", Transactions on High-Performance Embedded Architectures and Compilers IV, Springer-Verlag, Berlin, Heidelberg, 2011.

[10] S. Swanson, A. Schwerin, M. Mercaldi, A. Petersen, A. Putnam, Ken Michelson, M. Oskin, S. J. Eggers, "The WaveScalar architecture", ACM Transactions on Computer Systems (TOCS), v.25 n.2, p.1-54, May 2009.

[11] W. V. Kritikos , A. G. Schmidt , R. Sass , E. K. Anderson , M. French: "a programming model and on-chip network for multi-core systems on a programmable chip", International Journal of Reconfigurable Computing, 2012, p.2-2, January 2012.

[12] A.J.McAuley and P.Francis: "Fast routing table lookup using CAMs", IEEE INFOCOM 1993, pages 1382–1391, March 1993.

[13] A. Efthymiou and J. D. Garside: "An adaptive serial-parallel CAM architecture for low-power cache blocks", In Proc. of the ISLPED, pages 136–141, 2002.

[14] I. Mandal: "A low-power content-addressable memory (CAM) using pipelined search scheme", In *Proceedings of the International Conference and Workshop on Emerging Trends in Technology* (ICWET '10). ACM, New York, NY, USA, 2010, pp. 853-858. DOI=10.1145/1741906.1742103.