From the Dynamic Lattice Liquid Algorithm to the Dedicated Parallel Computer – mDLL Machine

Jarosław Jung[1], Rafał Kiełbik[2], Kamil Rudnicki[2], Krzysztof Hałagan[1], Piotr Polanowski[1], Andrzej Sikorski[3]*

[1]*Department of Molecular Physics, Technical University of Łódź*
90-924 Łódź, Poland

[2]*Department of Electronics, Technical University of Łódź*
90-924 Łódź, Poland

[3]*Department of Chemistry, University of Warsaw*
Pasteura 1, 02-093 Warsaw, Poland
**E-mail: sikorski@chem.uw.edu.pl*

Received: 13 November 2018; revised: 03 December 2018; accepted: 05 December 2018; published online: 24 December 2018

Abstract: The designing, production and testing of the mDLL machine led to the development of such a structure in which operational cells (e.g. KDLL) were located in the nodes of a three-dimensional torus network and the device was scalable. Thus, the future expansion of this device with additional Printed Circuit Boards (PCB) will not result in lengthened wire connections between Field-Programmable Gate Arrays (FPGA) or slow down the operation of the machine. The conducted tests confirmed the correctness of the adopted design assumptions and showed that by using mDLL one can effectively perform molecular simulations. Despite some structural shortcomings, the mDLL machine was a prototype that has already been sufficiently tested to allow the technology used in it to be used to build a device with a number of 1 million to 5 million KDLL cells. Such a device would already be suitable for simulating multi-particle systems with unprecedented speed.

Key words: Field Programmable Gate Array, topology of the network connections, parallel data processing, molecular simulations.

I. INTRODUCTION

Comprehensive computing systems such as computer clusters or supercomputers are very often used to simulate phenomena occurring in complex molecular systems [1-3]. However, the possibilities offered by computer simulations are subject to very important limitations. One of them is the small size of the simulation space. For smaller systems, this limitation results from finite memory resources and the number of simultaneously operating computing units. In the case of supercomputers that allow tests of objects containing a large number of elements, the limitation is related to the possibility of simulation in a “reasonable time” – that is, not exceeding a duration of a few to a dozen or so months of continuous system operation.

The paper presents the construction of a scalable mDLL machine containing operational cells (KDLL) [4] placed in the nodes of a face-centered cubic lattice (FCC) forming a three-dimensional torus [5]. The device is designed to perform molecular simulations using the dynamic lattice liquid algorithm (DLL) [6-7]. mDLL is a development of the concept of a DLL machine built earlier at the Łódź University of Technology.

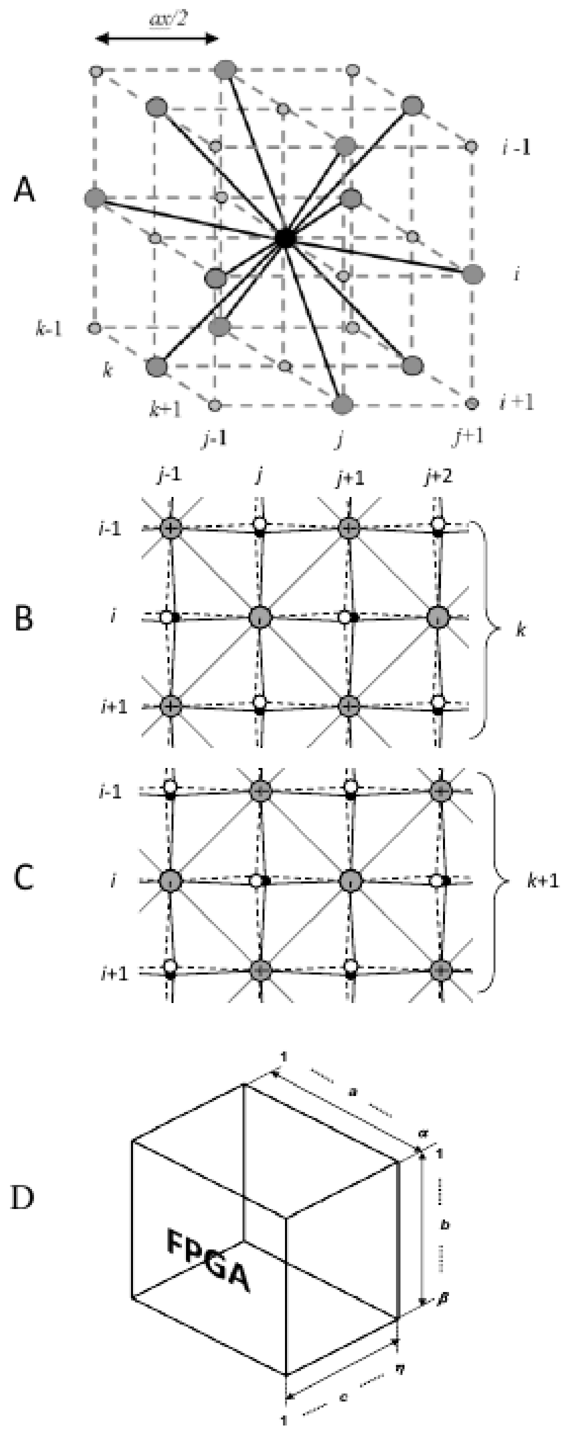


Fig. 1. The planes for the FCC network in which the KDLL operational cells in the DLL machine were placed along with the connection network. The planes in which the closest neighbors of the KDLL cell were located (A); network nodes lying in one plane, included in 3 integrated circuits U11, U12 and U13 (B); connections between KDLL cells (C); the colors are marked with the following planes: P1 – dark gray "behind the page", P2 – black 'in the plane of the page' and P3 – light gray 'before the card' (C); the numbering of network nodes of RCL in FPGA (D)

In the DLL machine described in Ref. [8-10], on 6 PCBs (in the Printed Circuit Board), three FPGAs (in the field-programmable gate array) were placed each containing 12 KDLL operational cells (Fig. 1B). KDLL cells implemented in FPGAs were located on virtual planes in nodes of the triangular network with coordination number 6 (Fig. 1A). For each implemented KDLL cell, 6 pins arranged on PCBs were assigned, and the remaining connections (2 times for 3 connections to KDLL cells placed on neighboring planes) were realized by means of cables (Fig. 1A and Fig. 1C). Together with the operating control systems, the DLL machine was a closed spatial structure.

However, the topological solution used in the DLL machine had the following significant disadvantages:

1. the machine was non-scaled – it could not be expanded by a larger number of KDLL cells,
2. the adopted data transmission system between KDLL cells, due to the limited number of soldering terminals in FPGAs, made it impossible to place more than a dozen KDLL cells in one FPGA chip,
3. it was necessary to make the configuration of FPGAs in three ways. This resulted from a different topology of the input-output lines for the three neighboring planes. Each of the planes was shifted relative to the neighboring one by one-half node of the triangular network (representation of planes P1, P2 and P3 in Figs. 1A or 1C). The numbering of network nodes of reduced cubic lattice (RCL) in FPGA is presented in Fig. 1D.

Based on the research already done on the DLL machine and the analysis of its future applications, three key conclusions have been drawn. These conclusions are the basis for taking the action to build a new device (with the working name mDLL). The conclusions were the following:

1. a scalable mDLL machine containing the same operating cells as in the DLL machine with KDLL placed in FCC nodes in a three-dimensional torus system has to be designed,
2. a new data exchange system between neighboring PCBs containing FPGAs has to be developed and tested because at least several dozen KDLL operational cells can be implemented in the latest FPGA chips,
3. at least one scalable module surrounded by 26 similar modules in order to check the new data exchange system has to be introduced,
4. PCBs have to be spatially fixed in such a way that the device, which will contain at least 1 million KDLL cells necessary to simulate the dynamics of molecules in complex molecular liquids can be created in the future.

II. SPATIAL STRUCTURE OF THE MDLL MACHINE

The work on the construction of the mDLL machine started with the analysis of the FCC network topology and the data exchange system between KDLL cells. It was assumed that the planes would intersect nodes in a different way than shown in Fig. 1. The FCC network nodes were every second node of the RCL, for which the distance between nodes was as $ax/2$, and the FCC network was as if "immersed" in RCL networks (Fig. 2A).

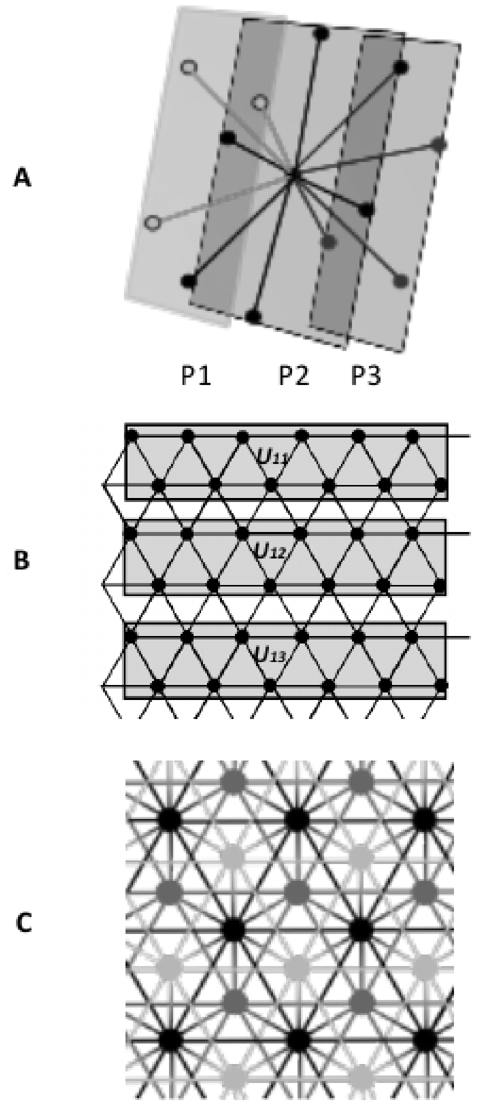


Fig. 2. Reduced cubic network RCL (A); straight plane (B); and turned (C). Indexes i, j, k indicate the location of nodes in the RCL network

The analysis of the FCC network fragment presented in Fig. 2A shows that two types of parallel planes could be distinguished, alternating with each other, on which the FCC network nodes are located. The first plane (hereinafter referred to as the straight plane) is spanned on 5 nodes with coordinates $(i-1, j+1, k)$, $(i+1, j+1, k)$, $(i+1, j-1, k)$, $(i-1, j-1, k)$ and (i, j, k). The second type of plane (hereinafter referred to as the rotated plane) is stretched on 4 nodes with coordinates $(i, j+1, k-1)$, $(i+1, j, k-1)$, $(i, j-1, k-1)$ and $(i-1, j, k-1)$ (the same plane contains points $(i, j+1, k+1)$, $(i+1, j, k+1)$, $(i, j-1, k+1)$ and $(i-1, j, k+1)$). Figs. 2B and 2C present the difference between straight and rotated planes. Figs. 2B and 2C show that the rotated plane is a straight plane offset by one node of the reduced RCL network. In the further part of the work, we call a straight plane in which the first point in the upper left corner ($i = 1$ and $j = 1$) will contain the FCC network node. Otherwise, a plane will be called the rotated plane.

KDLL cells in each FPGA system were assigned to positions in every second $\alpha \cdot \beta \cdot \eta$ of the virtual nodes (implemented in the FPGA system) of the RCL network (Fig. 3 and Tab. 3). For an even number of network nodes, the number of cells contained in the logic was $\alpha\beta\gamma/2$, and when the number was odd, the number of these cells was $(\alpha\beta\eta/2) - 1$ or $(\alpha\beta\eta/2) + 1$. In order to implement KDLL operational cells in FPGAs the topology of the RCL network inscribed in these systems had to be defined. It was also necessary to determine the number of lines needed for the data exchange between neighboring FPGAs in specific space directions. Depending on whether the number of rows β and columns α were even or odd, there were 4 pairs of complementary systems with different communication directions [9, 11]. These systems were arranged alternately on the plane and constituted pairs, hereinafter referred to as complementary pairs, in which systems marked with the letter P were called basic ones, and the U systems were called complementary ones.

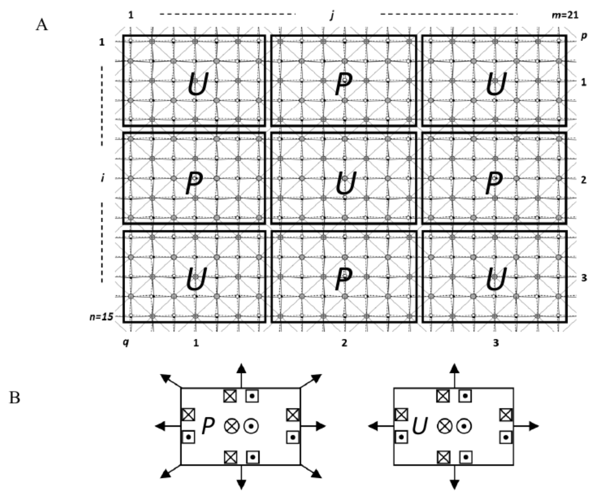


Fig. 3. The network of 157 cells placed in every second RCL network's node. The letter P denotes basic circuit, and the letter U supplementary circuit (A). The directions of communication for the complementary system (basic and supplementary planes). Wheels with cross indicate the direction of backward communication ("behind a paper sheet"), and circles with a dot indicate the forward direction ('before the paper sheet'); squares with cross or dot are the backward and forward oblique directions, respectively (B)

Fig. 3 shows an example of the arrangement of 3 surface fragments: two straight lines and one rotated line covered by one FPGA system containing 30 operational cells located at the reduced ZSK network nodes. The system contains $\alpha = 4$ columns, $\beta = 5$ rows and $\eta = 3$ levels. On the planes marked with $k - 1$ and $k + 1$ indices there are layers P1 and P3 of the FPGA chip, in which the directions of communication with neighboring cells in these planes correspond to the basic layout for α even and β odd.

In the third dimension (along the z axis), as well as for two dimensions (in the x, y plane), there are 4 possibilities of mutual arranging of the planes (depending on the parity α and β) in which there are pairs of straight and rotated planes. For each combination of plane pairs, they are always in front of each other the basic layout for α even and β odd (Tab. 1). In the selected direction the number of these lines, marked with symbol, $L_{x,y,z}^{X,Y,Z}$, is:

$$ {}_{\alpha,\beta,\eta}^{\;\;\Omega}L_{x,y,z}^{X,Y,Z}\,(i,j,k) = \Pi \cdot \Theta \tag{1}$$

where:

1. X, Y, Z define spatial directions and take binary values (e.g., $X = 1$ and $\bar{X} = 0$ – data sent, or $X = 0$ and $\bar{X} = 1$ – data not sent in the direction of the X axis),
2. x, y and z (divalent variables: -1 or 1) inform in which direction data is sent/received along the X, Y and Z directions (see inset to Fig. 4),
3. indices i and j are line and column numbers, and the index k is the number of the plane in which the FPGA is located (Fig. 3),
4. Ω is equal to 1 when the device is a simple machine (FPGA is in the upper left corner of the first plane for $i = 1$, $j = 1$ and $k = 1$ or Ω is equal to -1 for the rotated machine (when the upper left corner of the first plane does not contain a logic).
5. α, β, η indicate the number of ZSK network nodes included in the FPGA system counted in the x, y and z axis directions (Fig. 3).

The six-dimensional, linear vector Π defines the directions in space and is given by the formula:

$$\Pi = \begin{bmatrix} X\overline{Y}\overline{Z} & \overline{X}Y\overline{Z} & \overline{X}\overline{Y}Z & XY\overline{Z} & \overline{X}YZ & X\overline{Y}Z \end{bmatrix} \tag{2}$$

A six-dimensional, columnar vector Θ, whose elements depend on the number of nodes (α, β, η) of the RCL network contained in the FPGA and on the parameters' parity, α, β and η is described by the equation:

$$\Theta = \begin{bmatrix} 2\eta\beta - \eta - \beta \\ 2\alpha\eta - \alpha - \eta \\ 2\alpha\beta - \alpha - \beta \\ \frac{1}{2}\left[\eta - \Omega \underset{\leftarrow}{\eta}\underset{\leftrightarrow}{k}\left(\underset{\leftarrow}{\beta}\underset{\leftrightarrow}{i} + y\underset{\rightarrow}{\beta}\right)\left(\underset{\leftarrow}{\alpha}\underset{\leftrightarrow}{j} + x\underset{\rightarrow}{\alpha}\right)\right] \\ \frac{1}{2}\left[\alpha - \Omega \underset{\leftarrow}{\alpha}\underset{\leftrightarrow}{j}\left(\underset{\leftarrow}{\eta}\underset{\leftrightarrow}{k} + z\underset{\rightarrow}{\eta}\right)\left(\underset{\leftarrow}{\beta}\underset{\leftrightarrow}{i} + y\underset{\rightarrow}{\beta}\right)\right] \\ \frac{1}{2}\left[\beta - \Omega \underset{\leftarrow}{\beta}\underset{\leftrightarrow}{i}\left(\underset{\leftarrow}{\alpha}\underset{\leftrightarrow}{j} + x\underset{\rightarrow}{\alpha}\right)\left(\underset{\leftarrow}{\eta}\underset{\leftrightarrow}{k} + z\underset{\rightarrow}{\eta}\right)\right] \end{bmatrix} \tag{3}$$

The variables highlighted by arrows pointing left, right or in both two sides correspond to natural numbers s and they take the values -1, 0 and 1 according to the following definition:

$$\underset{\leftarrow}{s} = \frac{1-(-1)^s}{2}; \quad \underset{\rightarrow}{s} = \frac{1+(-1)^s}{2}; \quad \underset{\leftrightarrow}{s} = (-1)^s \tag{4}$$

The DLL and mDLL machines were designed in such a way that FPGAs simultaneously receive and send data to

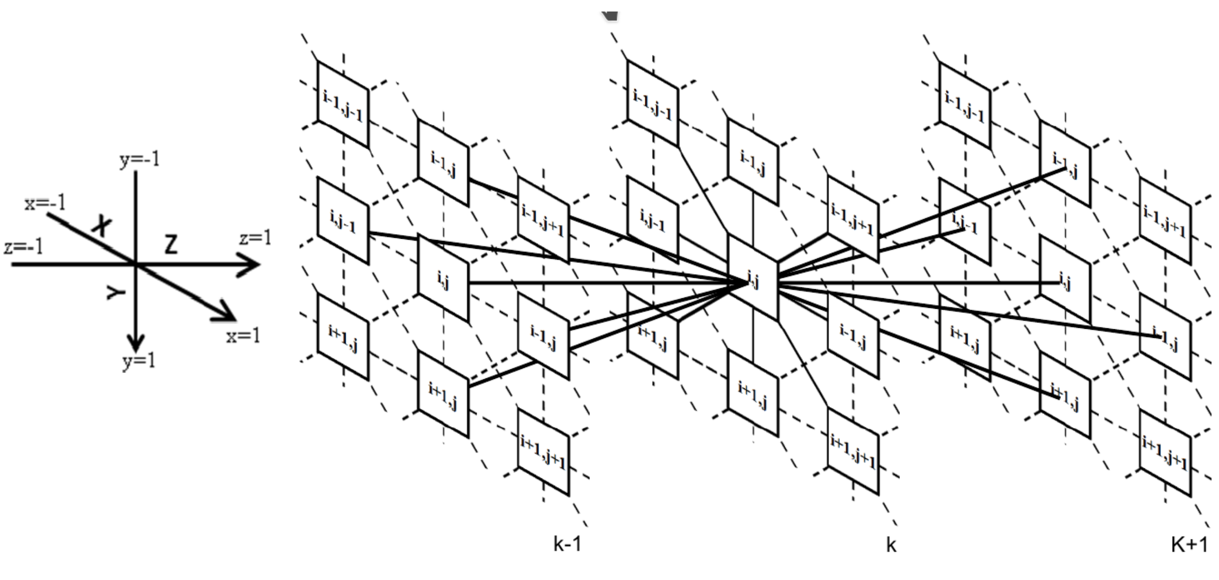


Fig. 4. Directions of data exchange for one FPGA. The insert determines the orientation of the coordinate axes in space

neighbors and, therefore, two signal lines are assigned to each direction. Thus, the number of pins from each FPGA integrated circuit (number of solder terminals) needed to connect KDLA cells to neighboring cells in other FPGAs on neighboring PCBMs is equal to $2 \cdot L^{X,Y,Z}_{x,y,z}$. Eq. (1) determines the number of these leads. Tab. 2 presents the results of calculations for 12 cells implemented in one FPGA ($\alpha = 12$, $\beta = 2$, $\eta = 1$) (as it was done in the DLL machine described in [8-9]) and for the case of 100 cells implemented in one layout. The data presented in Tab. 2 show that the number of pins and the number of signal wires per one FPGA system containing 12 cells enables the execution of a machine with a connection topology similar to that in the DLL machine [8-9]. However, if the number of implemented cells reaches the value of 100 or more, then it is impossible to build a machine in which the cells are connected directly via signal lines. This is due to two factors:

1. the number of needed leads would be greater than the number of available soldering tips in one integrated circuit (e.g., OSERDES and ISERDES blocks available in FPGA systems, which allow for LVDS transmission (in Low-Voltage Differential Signaling)),
2. the number of cables distributed between the plates would be too great for the machine to operate at a satisfactory level of reliability.

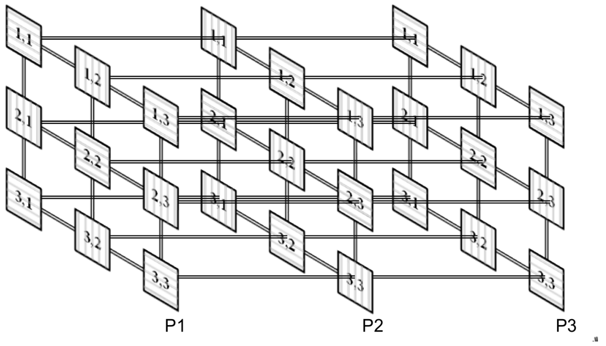
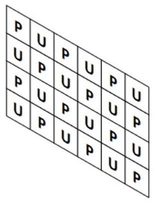
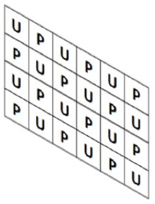
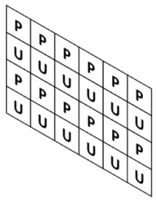
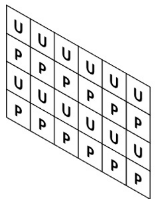
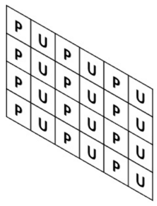
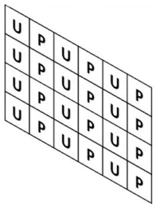
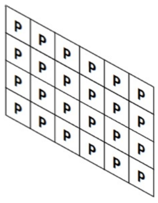
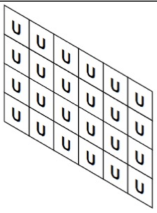


Fig. 5. The reduced communication channels between FPGAs

Tab. 1. Four possibilities for the arrangement of planes (depending on the parity of α or β), in which there are pairs of regular and rotated planes

α	β	*regular* plane	*rotated* plane
odd	odd		
even	odd		
odd	even		
even	even		

Tab. 2. An example of calculations for the number of bits of information grouped into 6 communication channels designed for serial data transfer to operation cells (bit by bit) implemented within neighboring FPGAs

The direction of data exechange	Direction of communication channels	The number of direction lines for FPGA	
		$\alpha.\beta \cdot \eta/2 \;=\; \mathbf{12}$	$\alpha.\beta \cdot \eta/2 \;=\; \mathbf{100}$
$X\overline{YZ}$	$x = 1$	${}_{12,2,1}^{\;1}L_{1,y,z}^{1,0,0}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{1,y,z}^{1,0,0}\,(3,3,3) = 67$
	$x = -1$	${}_{12,2,1}^{\;1}L_{-1,y,z}^{1,0,0}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{-1,y,z}^{1,0,0}\,(3,3,3) = 67$
$\overline{X}Y\overline{Z}$	$x = 1$	${}_{12,2,1}^{\;1}L_{x,1,z}^{1,0,0}\,(3,3,3) = 11$	${}_{5,5,8}^{\;1}L_{x,1,z}^{1,0,0}\,(3,3,3) = 67$
	$x = -1$	${}_{12,2,1}^{\;1}L_{x,-1,z}^{1,0,0}\,(3,3,3) = 11$	${}_{5,5,8}^{\;1}L_{x,-1,z}^{1,0,0}\,(3,3,3) = 67$
$\overline{XY}Z$	$x = 1$	${}_{12,2,1}^{\;1}L_{x,y,1}^{0,0,1}\,(3,3,3) = 34$	${}_{5,5,8}^{\;1}L_{x,y,1}^{0,0,1}\,(3,3,3) = 40$
	$x = -1$	${}_{12,2,1}^{\;1}L_{x,y,-1}^{0,0,1}\,(3,3,3) = 34$	${}_{5,5,8}^{\;1}L_{x,y,-1}^{0,0,1}\,(3,3,3) = 40$
$XY\overline{Z}$	$x = 1, y = -1$	${}_{12,2,1}^{\;1}L_{1,-1,z}^{1,1,0}\,(3,3,3) = 0$	${}_{5,5,8}^{\;1}L_{1,-1,z}^{1,1,0}\,(3,3,3) = 4$
	$x = -1, x = 1$	${}_{12,2,1}^{\;1}L_{-1,1,z}^{1,1,0}\,(3,3,3) = 0$	${}_{5,5,8}^{\;1}L_{-1,1,z}^{1,1,0}\,(3,3,3) = 4$
	$x = 1, y = 1$	${}_{12,2,1}^{\;1}L_{1,1,z}^{1,1,0}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{1,1,z}^{1,1,0}\,(3,3,3) = 4$
	$x = -1, x = -1$	${}_{12,2,1}^{\;1}L_{-1,-1,z}^{1,1,0}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{-1,-1,z}^{1,1,0}\,(3,3,3) = 4$
$\overline{X}YZ$	$x = 1, x = -1$	${}_{12,2,1}^{\;1}L_{x,1,-1}^{0,1,1}\,(3,3,3) = 6$	${}_{5,5,8}^{\;1}L_{x,1,-1}^{0,1,1}\,(3,3,3) = 3$
	$x = -1, x = 1$	${}_{12,2,1}^{\;1}L_{x,-1,1}^{0,1,1}\,(3,3,3) = 6$	${}_{5,5,8}^{\;1}L_{x,-1,1}^{0,1,1}\,(3,3,3) = 2$
	$x = 1, z = 1$	${}_{12,2,1}^{\;1}L_{x,1,1}^{0,1,1}\,(3,3,3) = 6$	${}_{5,5,8}^{\;1}L_{x,1,1}^{0,1,1}\,(3,3,3) = 2$
	$x = -1, x = -1$	${}_{12,2,1}^{\;1}L_{x,-1,-1}^{1,0,1}\,(3,3,3) = 6$	${}_{5,5,8}^{\;1}L_{x,-1,-1}^{1,0,1}\,(3,3,3) = 3$
$X\overline{Y}Z$	$x = 1, x = -1$	${}_{12,2,1}^{\;1}L_{1,y,-1}^{0,1,1}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{1,y,-1}^{0,1,1}\,(3,3,3) = 3$
	$x = -1, x = 1$	${}_{12,2,1}^{\;1}L_{-1,y,1}^{1,0,1}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{-1,y,1}^{1,0,1}\,(3,3,3) = 2$
	$x = 1, z = 1$	${}_{12,2,1}^{\;1}L_{1,y,1}^{1,0,1}\,(3,3,3) = 1$	${}_{5,5,8}^{\;1}L_{1,y,1}^{1,0,1}\,(3,3,3) = 2$

The formula (1) shows that the number of communication lines depends on the number of RCL nodes implemented in FPGAs. However, due to the mDLL scalability the number of wires needed for signal transmission should always be the same, regardless of the number of communicating KDLL cells implemented in FPGA chips. This was achieved by data compensation and reduction of 18 directions determined by communication lines (Fig. 4) to 6 directions determined by cubic network lines. The data were collected in packets, and then sent to neighboring FPGAs with high-speed serial connections via 6 bidirectional communication channels (Fig. 5). A detailed description of data collected in packets and their sending and receiving can be found in the document [11].

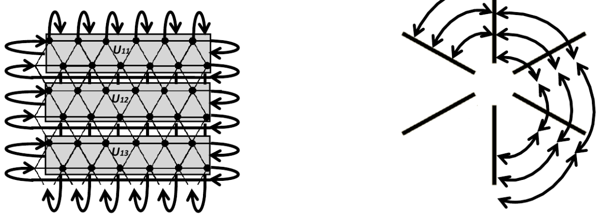


Fig. 6. The cyclic boundary conditions in the microDLL machine

The next issue was the development of an appropriate layout topology for FPGAs. In molecular simulations the spatial area available to molecules in the machine is limited. In most cases the systems analyzed contain large numbers of molecules and thus we want the boundaries of the system to be located in the infinity. In real systems it is not possible to build a network of infinite size and, therefore, in parallel machines, in which a large number of system operational cells (e.g., microprocessors) operate concurrently over time, n-dimensional pseudo-finite networks were used [12-14]. In the DLL machine [8-10], KDLL cells were placed in the nodes of a three-dimensional torus, which is formed after connecting the outermost elements of the network (Fig. 6). However, this solution precludes building a scalable machine, because adding more FPGAs will elongate the length of the connections of the extreme elements of the network will elongate the connection and for large systems it can exceed several meters. These connections would then be the slowest element of the entire machine, making it too slow to perform simulations of systems containing a large number of elements, i.e. of an order of 10^6 or larger.

In the described mDLL machine the scalability was obtained thanks to the technique called "leap frog" [5], which consists in alternating connection of integrated circuits located in the nodes of the three-dimensional torus network.

On the market there are cases in the form of metal cabinets adapted for joining together PCBs containing integrated circuits or microprocessors. However, the use of such cabinets for the construction of the mDLL machine would violate its spatial construction, in which PCBs with FPGA chips with FPGAs are mounted on flat, vertically positioned panels. Thus, a different method of the arrangement of FPGAs has been developed, which does not change the spatial structure of the machine and at the same time uses a system of alternating connection of integrated circuits (details of this solution can be found in the patent specifications [15-16]).

III. DESIGN ASSUMPTIONS OF THE MDLL MACHINE

The mDLL machine was built according to the following assumptions:

1. in FPGAs a virtual reduced cubic network RCL with number of nodes $\alpha\, \beta\, \eta$ (with $\alpha = \beta$) was implemented, and in every second node of this network KDLL operational cells were placed,
2. the position of KDLL cells in the RCL network configured in FPGAs was determined by indexes a, b, c (index $a = 1 \ldots \alpha$ corresponds to the x coordinate, index $b = 1 \ldots \alpha$ corresponds to the y coordinate, and the index $c = 1 \ldots \eta$ corresponds to the coordinate z),
3. the device consists of $\delta = 3$ panels containing $n = 3$ rows (counted in the direction of the y axis) and $m = 3$ columns (counted in the direction of the x axis),
4. panel numbers are marked with the k index, with the panel marked by the number $k = 1$ being the straight plane,
5. on the panels, at the intersection of each column (numbered by the index i) and each row (numbered with the index j), PCBs containing 4 FPGAs “transferred” from nodes i, j; and, $2m - j + 1$; $2n$ and $+12m - j + 1$ and $2n - i + 1,\ j$ of a virtual cubic network were placed. The nodes of this network cover the areas occupied by $\alpha.\beta$ RCL nodes implemented in individual FPGAs. These areas constitute the subnet of the full RCL network of the entire panel with $2m.\alpha$ rows and $2n.\beta$ columns [17]. Four FPGAs belonging to one panel were marked with indices $s = 1$, $s = 2$, $s = 3$, $s = 4$, respectively.

For a machine defined in this way, a series of equations allowing for describing its functionality and spatial construction were formulated. It turned out that the system of equations used for converting virtual spatial coordinates $x,\ y,\ z$ operational cells implemented in FPGAs to their x_s, y_s, z_s coordinates in the actual RCL network was found most useful in practice. The equation used to calculate coordinates $x,\ y,\ z$ of the KDLL operational cell, whose position in the FPGA set indices $a,\ b,\ c$, placed in position s on the PCB marked with indices $i,\ j$ on the k-th panel has the following form:

$$(x, y, z) = \omega \cdot \Gamma \cdot (x_s, y_s, z_s) \tag{5}$$

where the coordinate values x_s, y_s, z_s are equal to:

$$\begin{aligned} x_s &= \lambda^s_{11}\left[(j-1)\cdot\alpha + a\right] + \lambda^s_{12}\,(2m+1) \\ y_s &= \lambda^s_{21}\left[(i-1)\cdot\alpha + b\right] + \lambda^s_{22}\,(2n+1)\,, (k-1) \\ z_s &= (k-1)\,\beta + \lambda^s_{31}c + \lambda^s_{32}\,(\eta+1) \end{aligned} \tag{6}$$

The elements of the λ^s_{pq} matrix can take values of 0, 1 or -1 and are expressed by the following formula:

$$\begin{bmatrix} \lambda^s_{11} & \lambda^s_{12} \\ \lambda^s_{21} & \lambda^s_{22} \\ \lambda^s_{31} & \lambda^s_{32} \end{bmatrix} = \\ = \begin{bmatrix} (s-2)\cdot\left(\underset{\rightarrow}{s} - \underset{\leftarrow}{s}\right) - \underset{\rightarrow}{s} & \frac{1}{2}\left(4\underset{\rightarrow}{s} + 2\underset{\leftarrow}{s}\,s - s - \underset{\leftarrow}{s}\right) \\ 2\underset{\leftarrow}{s} - s + 3\underset{\rightarrow}{s} & \frac{1}{2}\left(s - \underset{\leftarrow}{s} - 2\underset{\rightarrow}{s}\right) \\ \underset{\leftarrow}{s} - \underset{\rightarrow}{s} & \underset{\rightarrow}{s} \end{bmatrix} \tag{7}$$

where s parameters are defined in eq. (4).

The size of Γ is equal to 0 when there is no operational cell in the node of the RCL network, and equal to 1 when the cell occupies a place in the node. Γ is defined by the following formula:

$$\begin{aligned} \Gamma = \underset{\rightarrow}{\eta} \cdot \Big[&\underset{\rightarrow}{\alpha} \cdot \underrightarrow{(A+B+C)} \\ &+\underset{\leftarrow}{\alpha} \cdot \underrightarrow{(A+B+C+I+J)}\Big] \\ &+\underset{\leftarrow}{\eta} \cdot \Big[\underset{\leftarrow}{\alpha} \cdot \underleftarrow{(A+B+C+k)} \\ &+\underset{\rightarrow}{\alpha} \cdot \underleftarrow{(A+B+C+I+J+k)}\Big] \end{aligned} \tag{8}$$

Variables A, B, C, I, J correspond to indices a, b, c, i, j converted from the system with cyclic boundary conditions to the open system (the cube containing the RCL network) and can be determined as:

$$\begin{aligned} A &= \lambda^s_{11}a + \lambda^s_{12}\,(\alpha+1) & J &= \lambda^s_{11}j + \lambda^s_{12}\,(2m+1) \\ B &= \lambda^s_{21}b + \lambda^s_{22}\,(\alpha+1) & I &= \lambda^s_{21}i + \lambda^s_{22}\,(2n+1) \\ C &= \lambda^s_{31}c + \lambda^s_{32}\,(\eta+1) & & \end{aligned} \tag{9}$$

Coefficient ω is equal to:

$$\omega = \frac{\underset{\rightarrow}{\eta} + \underset{\rightarrow}{\delta}}{1 + \underrightarrow{(\delta+\eta)}} \tag{10}$$

and is 0 when η and δ are odd (when it is impossible to meet cyclical boundary conditions). In other cases, $\omega = 1$. Eq. (5) determines the initial position of all elements of the simulated system and to track the trajectory of their motion.

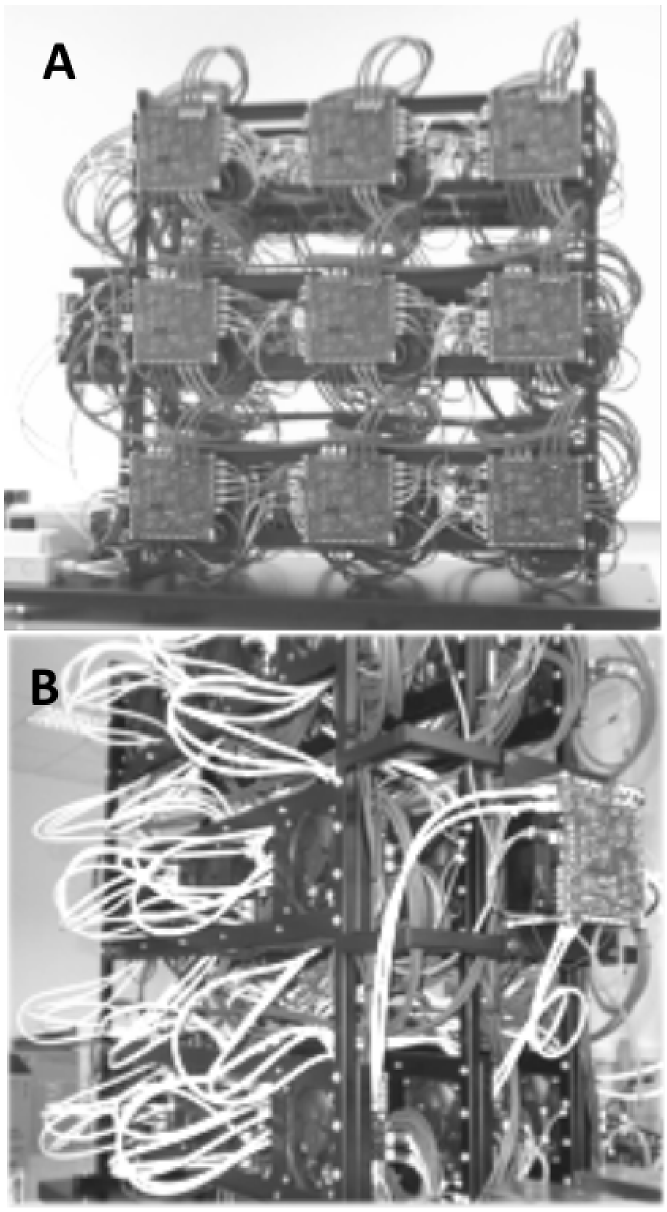


Fig. 7. The mDLL machine containing 27 PCBs placed on 3 panels (A) and the control system of the USDLL machine (B)

IV. ELECTRONIC IMPLEMENTATION

The mDLL device was made in such a way that it was possible to check the developed concepts of machine scalability and the method of sending information between neighboring operational cells. The mDLL machine with fastened PCBs on a steel truss and data transmission lines is shown in Fig. 7. It consists of three vertical, parallel-arranged panels, on which 9 PCBs are evenly spaced. The PCB, located in the central part of the central panel, was surrounded by similar 26 plates. With such a spatial construction, data exchange between the FPGAs belonging to the central board and the systems on neighboring boards had to be done only via physically mounted signal lines. This allowed testing the developed concept of serial data transfer. In 108 FPGAs, RCL networks containing 32 KDLL cells were implemented, which gave 3240 KDLL cells in the entire machine. All systems were powered by voltage converters, which were cooled by air using fans. A board with an additional FPGA chip used to coordinate the simultaneous operation of all cells (USDLL machine control unit [4, 8-9]) was located on the side wall of the device (Fig. 7B).

Data exchange between individual machine modules took place in three ways:

1. In order to minimize the duration of the transmission resulting from its serial nature, the transmission in the

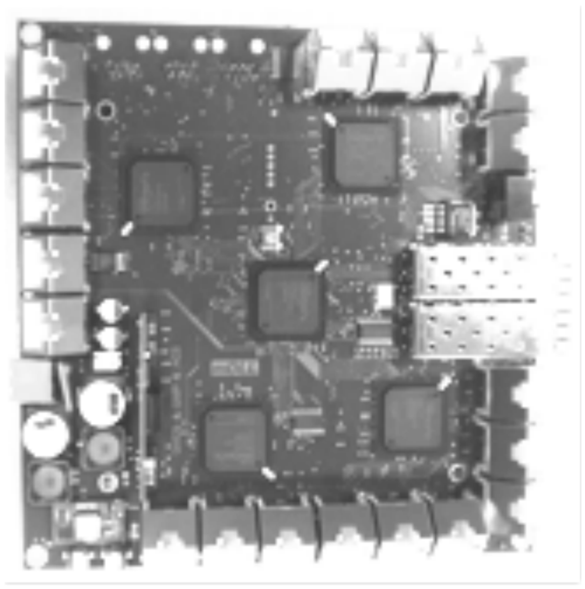


Fig. 8. PCB with FPGA chips. The central system was used to synchronize the work of the remaining 4, in which KDLL cells were implemented

LVDS standard at a speed of up to 800 Mb/s was used to send and receive data between operating cells. Ethernet cables were used to connect the systems belonging to different PCBs. In each of the 6 communication channels there were 6 "twisted pair", which allowed exchanging data between cells at 3.2 Gb/s.

2. For the purpose of synchronizing the machine's operation, requiring communication between the operational cells and the CKDLL control system, a mechanism of signal transmission via serially connected PCB boards was used. Two Ethernet cables were used, i.e. 8 twisted-pair cables, of which 5 twisted-pair cables were dedicated to "up to CKDLL" communication, and 3 twisted-pair cables were intended for communication in the 'from CKDLL' direction.
3. For the purpose of the configuration of FPGAs, machine initialization and acquisition of simulation results, fiber optic lines and 6 MGT (Multi-GigabitTransceiver) chips available in Spartan systems supporting transmission on these lines at up to 3.125 Gb/s in each direction were used. On each of the PCBs there are 5 FPGAs (catalog number XC6SLX75-FGG484-2), one of which was in their central location and served as the managing element for the other four (Fig. 8). This system has been equipped with EEPROM configuration memory (Electrically Erasable Programmable Read-Only Memory). Its contents at the moment of power supply was used to configure the central system in such a way that it could communicate with other similar devices on neighboring boards. A chain of series connected central circuits was created, the first of which was with the CKDLL system. Thanks to such a solution, from the PC computer connected to CKDLL, using the JTAG protocol, all FPGA systems in the machine were configured.

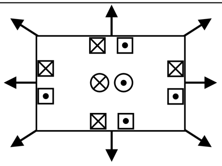
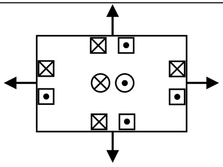
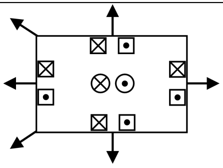
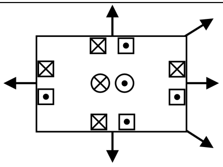
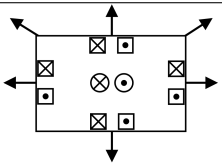
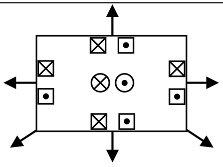
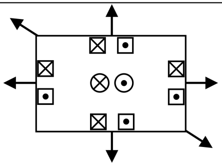
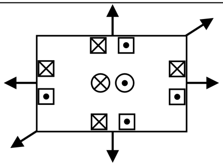
V. MDLL MACHINE OPERATION TESTS

The checking of the communication mechanisms between the FPGAs was an important part of the verification of the correctness of the mDLL machine operation. Two tests were done for this purpose. The first one consisted in self-verification of the neighbors with whom each of the FPGAs communicates. Each transmitter of any FPGA system sent a unique identifier that uniquely determines the position and direction of transmitting this transmitter. On the receiving side, the received identifiers were compared to the expected values and any deviations were reported as connection errors. In the second test the analysis of the motion of a virtual "ball" moving accidentally between KDLL cells located in FCC nodes within a three-dimensional torus was made. During this test, the number of visits to the "balls" in each node was remembered, and the expected result was an even distribution of these visits in all nodes of the network. The quality of this distribution was assessed by evaluating the spread of the number of visits φ calculated according to the formula:

$$\varphi = \frac{n_{\max} - n_{\min}}{n_{\max}} \qquad (11)$$

where $n_{\max}$ and $n_{\min}$ are the largest and the least observed number of visits at any of the nodes. The results of this tests are shown in Fig. 9. One can observe that φ decreases with the

Tab. 3. Four identified pairs of complementary systems for which the communication directions depend on whether the number of rows α and columns β are odd or even

Directional channels for pairs of *complementary* logic systems – 2D		The parity RCL network nodes counted in horizontal and vertical directions	
basic circuit $P = 1, U = 0$	*supplementary* circuit $P = 0, U = 1$	α	β
		odd	odd
		even	odd
		odd	even
		even	even

number of steps of the algorithm, which clearly proves that all connections between the nodes of the implemented FCC network were used to the same extent. The "ball" test was also an excellent way to verify the pseudo-random number generator used. It was carried out for different LFSR (Local Feedback Shift Register), and the obtained results were additionally compared with simulations carried out using a PC computer using the standard RAND function. The obtained results showed that the generator with the 64-bit LFSR register is as effective as the other solutions. It requires almost half as many hardware resources as its 128-bit counterpart. Additionally, taking into account the fact that each KDLL cell is equipped with its own generator, application in the final version of the 64-bit generator allowed reducing the number of logic modules needed to obtain the received implementation.

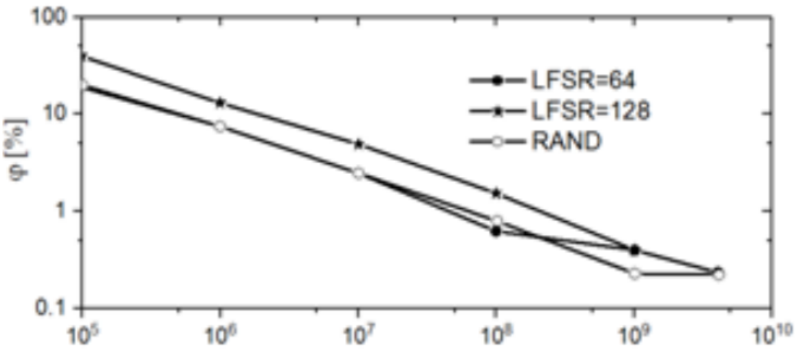


Fig. 9. The number of visits of nodes in the function of the number of steps for different methods of generating random numbers

Another test was checking the correctness and efficiency of the DLL algorithm. As we could not carry out of a comparison with another machine similar to mDLL, we decided to compare the speed between the implementation of a given simulation using mDLL and a PC. For this purpose, a specially prepared application allowing emulation of mDLL was started on the computer. This application was executed employing the same variant of the DLL algorithm as the one implemented in the hardware. The test simulations concerned the cooperative motion of objects (small molecules) at high densities, i.e. with a fully filled system and with no interactions (the athermal system). A computer with 2 GB of operating memory and a dual-core Intel Pentium D 3.4 GHz processor was used, with the reference application using only one of the processor cores. Analysis of the computational efficiency of the mDLL device showed that its advantage over a typical personal computer is at least 40 times. Due to the serial data exchange between modules, performing one simulation step for one liquid molecule required an average of twice as many system clock cycles as in the case of a reference sequential implementation running on a PC computer. The system clock in the prototype made was only 100 MHz, which was 34 times less than in the processor used for the tests, which in combination with 2 times more cycles gave almost 70 times the time to perform one step of simulation for one molecule. However, it should be noted that in contrast to the standard processor, simultaneous simulations for a system containing more than 3 000 molecules could be performed in mDLL.

The comparison of power consumption appeared to be as good as the computational efficiency. The assumption that a typical personal computer needs 200 W to power the motherboard, processor and operating memory in confrontation with the fact that the maximum power consumption of the prototype made is 1000 W, led to the conclusion that a 40-fold acceleration of calculations was paid for mDLL only a 5-fold increase in the demand for electricity.

VI. CONCLUSIONS

In this paper we described the process of designing, production and testing of a computer called mDLL machine. The idea of this machine was based on the Dynamic Lattice Liquid (DLL) idea – a model describing properties of liquid systems. The computer was designed in such a way that its future expansion with additional Printed Circuit Boards (PCB) would not result in lengthened wire connections between Field-Programmable Gate Arrays (FPGA) and therefore, it would not slow down the operation of the machine. Verification of the FPGA neighbors and test simulations were performed. They confirmed that the design assumptions were correct and mDLL appeared to be able to perform molecular simulations effectively. The mDLL machine can also be treated as a step towards the next device with a number of a few million of operational cells.

Acknowledgment

This work was supported by the Polish National Science Center (NCN) grant UMO-2017/25/B/ST5/01970.

References

[1] B.G. Fitch, A. Rayshubsky, M. Elftheriou, T.J.C. Ward, M.E. Giampapa, M.C. Pitman, J.W. Pitera, W.C. Swope, R.. Germain, Blue Matter: Scalling of N-body simulations to one atom per node, IBM J. Res & Dev, **52** 145 (2008).

[2] X. Hu, L. Hong, M.D. Smith, T. Neusius, X. Cheng, J.C. Smith J., *The Dynamics of Single Protein Molecules Is Non-Equilibrium and Self-Similar over Thirteen Decades in Time*, Nature Physics **12**, 171–174 (2016).

[3] W. Chen, E. De Schutter, *Parallel STEPS: Large Scale Stochastic Spatial Reaction-Diffusion Simulation with High Performance Computers*, Frontiers in Neuroinformatics, **11:13**, doi: 10.3389/fninf.2017.00013 (2017)

[4] P. Polanowski, *Implementacja sieciowych modeli cieczy i polimerów w symulacjach opartych na obliczeniach równoległych*, Rozprawa doktorska, Politechnika Łódzka, (2002).

[5] R.D. Jackson, A.J. Hoane Jr., *Modular Infinitely Extendable Three Dimensional Torus Packing Scheme for Parallel Processing*, US Patent no. US005715391A, http://www.google.ch/patents/US5715391 (1998).

[6] T. Pakula, *Collective dynamics in simple supercooled and polymer liquids*, J. Mol. Liq. **86**, 109-121 (2000).

[7] P. Polanowski, J.K. Jeszka, K. Krysiak, K. Matyjaszewski, Influence of intramolecular crosslinking on gelation in living copolymerization of monomer and divinyl cross-linker. Monte Carlo simulation studies, Polymer, **79**, 171-178 (2015).

[8] P. Polanowski, J. Jung, R. Kiełbik, A. Napieralski, K. Lichy, *Od algorytmu dynamicznej cieczy sieciowej do dedykowanego komputera równoległego*, Przegląd Elektrotechniczny, **84**, 69-73 (2008).

[9] J. Jung, *Od wybranych zagadnień elektroniki organicznej do uniwersalnej maszyny przeznaczonej do analizy zjawisk zachodzących w gęstych układach wieloskładnikowych*, Rozprawa habilitacyjna, Politechnika Łódzka (2016).

[10] J. Jung, P. Polanowski, R. Kiełbik, *Special state machine based on Dynamic Lattice Liquid model*, International Journal of Engineering Science and Innovative Technology (*IJESIT*), 3, 360-367 (2014).

[11] J. Jung, P. Polanowski, R. Kiełbik, K. Hałagan, W. Zatorski, J. Ulański, A. Napieralski, T. Pakuła, *A parallel machine with reduced number of connections between logical circuits*, (2016), EP Patent Application no. EP3079073A*1*, https://www.google.com/patents/EP3079073A1?cl=un (2016).

[12] G.G. Pachanek, N.P. Pitsianis, E.F. Barry, T.L. Drabenstott, *Method and Apparatus for Manifold Array Processing*, US Patent no. US6167502A, https://www.google.tl/patents/US6167502 (2000).

[13] H.W. Daniel, Parallel Processor, EP Patent no. EP0501524A2.

[14] www.google.com.na/patents/EP0501524A2?cl=un (1992).

[15] R.S. Passint, G. Thorson, M.B. Galles, *Hybrid Hypercube/Torus Architecture*, US Patent no. US6230252B1, http://www.google.ch/patents/US6230252 (2001).
[16] J. Jung, P. Polanowski, R. Kiełbik, K. Hałagan, W. Zatorski, J. Ulański, A. Napieralski, T. Pakuła, *Panel z układami elektronicznymi i zestaw paneli*, RP Patent no. P.405479, http://regserv.uprp.pl/register/application?number=P.405479 (2016).
[17] J. Jung, P. Polanowski, R. Kiełbik, K. Hałagan, W. Zatorski, J. Ulański, A. Napieralski, T. Pakuła, *A panel with electronic circuits and a set of panels*, EP Patent Application no. EP3079071A1, https://google.com/patents/EP3079071A1?cl=en (2016).
[18] J. Jung, P. Polanowski, R. Kiełbik, K. Hałagan, W. Zatorski, J. Ulański, A. Napieralski, T. Pakuła, *A parallel machine having operational cells located at nodes of a face centered lattice*, EP Patent Application no. EP3079072A1, https://www.google.com.na/patents/EP3079072A1?cl=en (2016).

Jarosław Jung received his M.Sc. degree in Physics in 1987 from the University of Lodz and in Electronics in 1990 from the Technical University of Lodz[A4]. He earned the PhD in Chemistry from the Technical University of Lodz[A5] in 2001 and received habilitation in Electronics in 2016. Currently, he is working in the Technical University of Lodz[A6] in the Department of Molecular Physics. His fields of interest cover research of organic semiconductors, construction of equipment for measurement of photoconduction in organic semiconductors and the construction of parallel processing machines that are designed to simulate dense molecular systems.

Rafał Kiełbik is an assistant professor at the Department of Microelectronics and Computer Science, Lodz University of Technology, Łódź, Poland. In 2005, he received his PhD degree for a dissertation on efficient methods of resource management in reconfigurable systems. His research focuses on inventing, developing and using reconfigurable devices for embedded systems as well as for general purpose and high performance computing.

Kamil Rudnicki is co-owner of Brightelligence Ltd, UK. He received his MSc from the Department of Microelectronics and Computer Science, Lodz University of Technology, Poland in 2008, and PhD from the School of Engineering, University of Glasgow, United Kingdom, in 2014. He performs research in the field of programmable devices with applications in embedded systems and high performance computing. His scientific background helps him succeed in challenging commercial projects. In his free time he enjoys cycling and playing Rummikub.

Krzysztof Hałagan received his M.Phil in physics in 2006 and PhD degree in chemical technology in 2013, both from Lodz University of Technology. His main fields of interest are computer simulations of molecular systems, including application of the Dynamic Lattice Liquid algorithm to various physicochemical problems (like diffusive behaviour in complex liquids and polymerization kinetics), and hardware-related issues like dedicated computing devices.

Piotr Polanowski received his MSc degree in physics in 1987 from the University of Łódź. He earned the PhD in Chemistry from the Lodz University of Technology in 2002 and habilitation (D.Sc.) in Physics from the Adam Mickiewicz University of Poznań. He is currently working in the Lodz University of Technology in the Department of Molecular Physics. His fields of interest cover simulations of complex molecular and macromolecular systems with proper dynamic behavior, parallel computing (hardware and software) in application to complex molecular systems, and simulation software development.

Andrzej Sikorski graduated from the Department of Chemistry, University of Warsaw. He completed his PhD thesis in Chemistry in 1986 in the same Department. In 1987/89 and 1990/91, he worked as a postdoc and visiting professor in Washington University in St. Louis and in The Scripps Research Institute in La Jolla. In 1995, he obtained his DSC degree in Chemistry. His main scientific interest are: theory and simulations of polymers at interfaces and in confined geometries and molecular transport in a crowded environment.