# Optimized Implementation of Lattice Boltzmann Method in ARUZ

Grzegorz Jabłoński and Joanna Kupis

***Abstract*—The paper presents the optimized implementation of the Lattice Boltzmann method on ARUZ, a massively parallel FPGA-based simulator located in Lodz, Poland. Compared to previous publications, a performance improvement of 46% has been achieved on D2Q9 lattice due to overlapping of communication with computation. The presented approach is suitable also for other cellular automata-based simulations. Extrapolation of results from the single ARUZ board suggests, that LBM simulation of 1080 $\times$ 480 lattice on 18 panels of ARUZ would reach the performance of $302 \cdot 10^3$ MLUPS (Million Lattice Updates per Second). This implementation has been compared to the classical supercomputer solution, giving much better power efficiency (3000 MLUPS/kW vs. 1280 MLUPS/kW, respectively).**

***Index Terms*—Distributed System, Reconfigurable System, FPGA, Lattice Boltzmann Method, ARUZ.**

## I. Introduction

ARUZ (Analizator Rzeczywistych Układów Złożonych, Analyser of Real Complex Systems) is a massively parallel FPGA-based simulator located at Lodz Technopark. This machine has been designed for execution of a single algorithm (Dynamic Lattice Liquid — DLL [1]) in mind [2] [3] [4]. Recently an implementation of the Lattice Boltzmann method [5] on ARUZ has been presented [6]. This paper presents an improvement to this implementation.

## II. The ARUZ Architecture

The ARUZ [6] is composed out of 25920 Field Programmable Gate Arrays (FPGAs), interconnected by 70 000 twisted-pair cables, composed of 20 panels, 18 usually engaged in ongoing simulations, whereas remaining two redundant used as a standby in case of technical problems. Every panel consists of 12 rows, each containing 12 PCBs, called DBoards (Daughter Boards). In total, there are 144 DBoards in each panel, giving 2880 for the whole machine.

Each simulation board carries 9 FPGAs: 8 of them called DSlaves (Artix XC7A200T), constitute the resources for execution of the simulation algorithm and the remaining one called DMaster (Zynq XC7Z015) manages the operation of DSlaves. The DMasters are responsible for configuring DSlaves, determining the current state of the whole system, initializing simulation process and archiving its results and are connected via 1 Gb Ethernet-based global communication to a PC controlling the machine.

G. Jabłoński and J. Kupis are with the Department of Microelectronics and Computer Science, Lodz University of Technology, ul. Wólczańska 221/223, 90-924 Lodz, Poland (e-mail: gwj@dmcs.p.lodz.pl)

This work was supported by the Polish National Science Centre grant 2015/19/N/ST6/01191. Tests on Prometheus were supported in part by the PLGrid Infrastructure.

Each of DSlaves is equipped with the communication interfaces to DMaster and the closest neighboring FPGAs in a 3D simulation space. This local communication enables the dataflow between DSlaves and the data exchange between DMaster and DSlaves within one DBoard in order to initialize simulation cells and read their state during the simulation process. Local communication is bi-directional, LVDS-based and uses 1 GHz source-synchronous clock and a single data line in each direction. It is used for communication between FPGAs placed on the same board, on the same panel or on the neighboring panels. The reception of data is confirmed by acknowledgement packets and the integrity of transmitted messages is protected by the 32-bit CRC. In case of errors, the packets are retransmitted. There are 11 connections from every DSlave: 4 to neighboring boards on the same panel (up, down, left, right), 2 to the neighboring panels (front, rear), 4 to the other DSlaves on the same DBoard (L1, L2, L3, L4) and one to the DMaster (M). The connections between boards are established using Cat6 STP cables. The longest cables, between neighbouring panels, have the length of ca. 4 meters. This interconnection network allows creating of 3D, 2D or 1D simulation grids of different sizes. For 2D simulations, the interconnection structure with an aspect ratio closest to one has the x-axis going along all the panels. As a result, we have a matrix of 18 (panels) $\times$ 12 (board/panel) = 216 boards in the x axis and 12 boards in the y axis.

## III. The Lattice Boltzmann Method

The lattice Boltzmann method (LBM) was proposed in 1988 in [5]. It is dedicated for solving the Navier-Stokes equations describing fluid dynamics. The method originates from the earlier lattice gas automata model [7], but differs from its predecessor in operating at mesoscopic level, i.e. by using distribution functions instead of velocities of individual molecules.

The basis of the LBM is two- or three-dimensional regular grid. Each of the grid nodes has several associated state variables $f_i(\vec{x}, t)$. These variables constitute the density of molecules at place $\vec{x}$ and time $t$ moving in direction $\vec{c_i}$. There are many variants of lattices applied in this model. The most popular one for 2D simulations is the D2Q9 lattice presented in Fig. 1.

*ISSN 2080-8755*

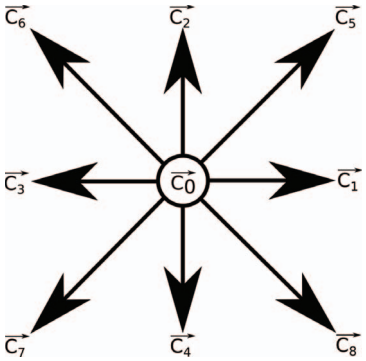



Figure 1. The D2Q9 lattice

The macroscopic quantities – molecule density and velocity at the given node – can be computed from $f_i(\vec{x}, t)$.

The lattice Boltzmann algorithm operates in two main phases:

- **Streaming:** the distribution functions $f_i$ are propagated to the neighboring nodes in the direction $\vec{c_i}$. No computations are performed during this phase.
- **Collision:** The particles incoming from all directions to a given node collide with one another, creating a new distribution of $f_i$. This phase contains many floating-point operations.

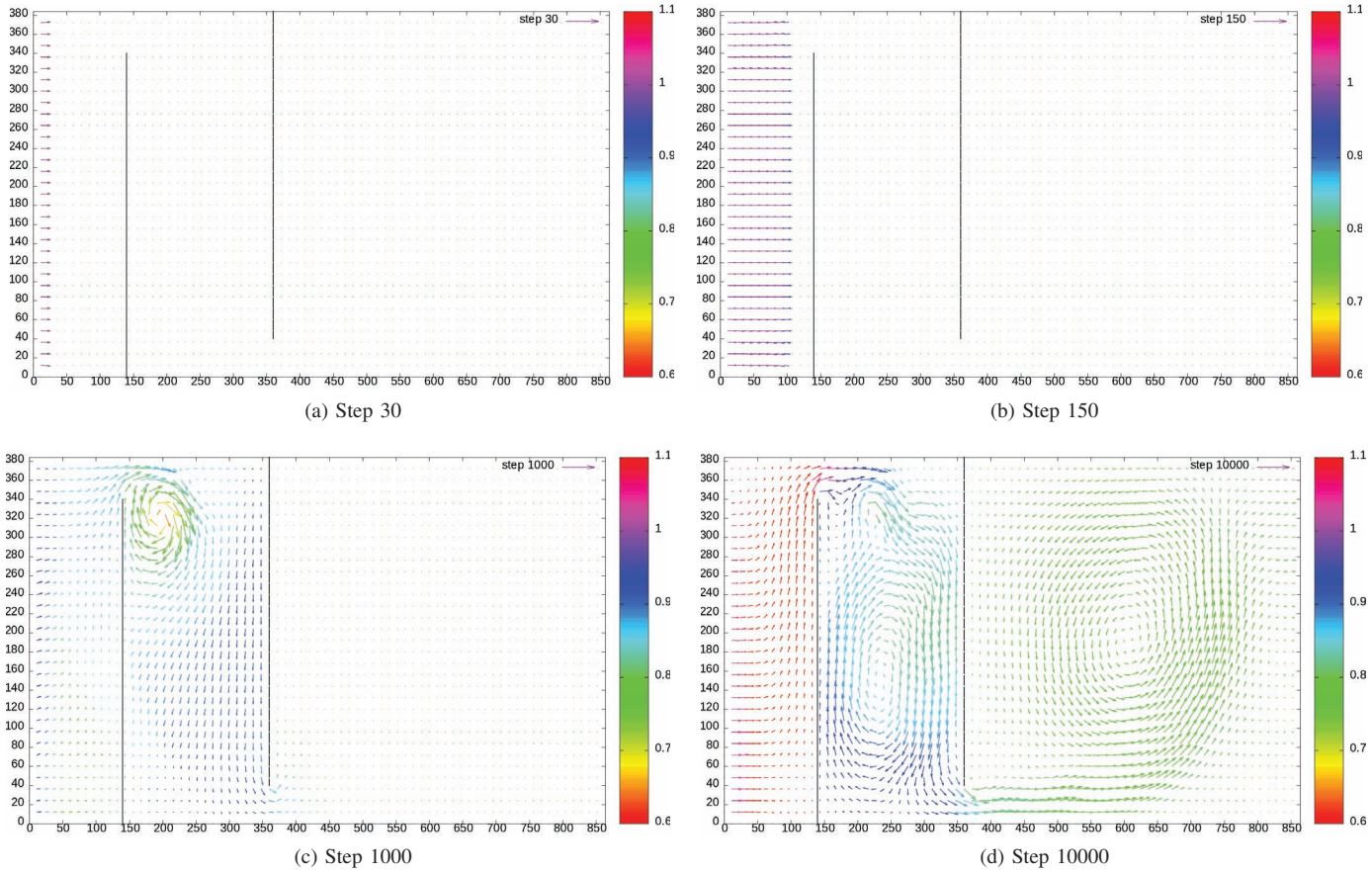
The lattice Boltzmann method operates locally in the collision phase and requires sending a single floating point value of a distribution function to each of the neighboring nodes during the streaming phase therefore it fits perfectly to the architecture of ARUZ.

Figure 2 presents the results of example simulation using the D2Q9 variant of the lattice Boltzmann method [6].

## IV. FPGA Implementation of the LBM Algorithm

To implement the simulation, all the DSlaves in the ARUZ are connected into the two-dimensional computational grid. Each of the DSlaves implements a small portion of the computational domain and is connected to at most 4 neighboring FPGAs. The implementation of the streaming phase of the LBM is quite straightforward, as it only requires the data movement inside of between neighboring FPGAs. If the entire simulation domain could fit inside one FPGA, the streaming phase would take just one clock cycle. The collision phase is much more complex as it requires about 90 floating-point operations. As the boundary conditions require special treatment, the LBM simulation in [6] requires 7 different collision operators.

The internal structure of the DSlave is presented in Fig. 3. Apart from the control circuit, the FPGA contains 11 transceiver modules (in a 2D simulation only 5 are actually used - one to the DMaster and 4 to the neigbouring nodes, depending on the specific location of the FPGA) and the matrix of $N \times N$ computational modules, implementing the collision




(a) Step 30

(b) Step 150

(c) Step 1000

(d) Step 10000

Figure 2. Example fluid flow simulation on 864 × 384 lattice at 4 different time steps (performed on a PC). Arrow size and direction - fluid velocity, arrow color - fluid density.

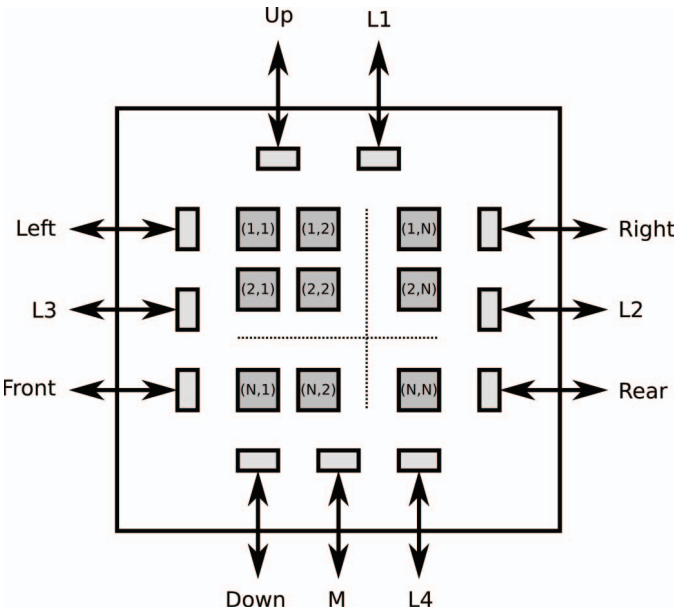



Figure 3. Internal structure of DSlave

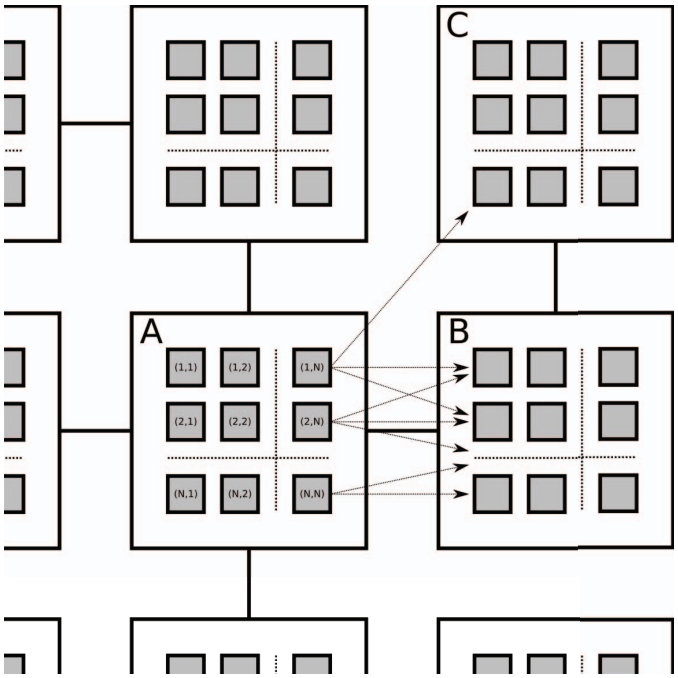



Figure 4. Data exchange between neighbouring FPGAs

operators and synthesized from the C source code with the application of Vivado HLS 2017.2. As there are no diagonal connections between neighboring FPGAs, the streaming phase is divided into 2 sub-phases. In the sub-phase 1, $3 \times N - 1$ single-precision floating-point numbers are sent from FPGA A to FPGA B (see 4), in the sub-phase 2 one number is sent from FPGA B to FPGA C. Similar transmissions occur simultaneously over 3 remaining links in upward, leftward and downward direction. The HLS estimations for all the function variants are presented in Table I suggesting, that it is possible to fit a 4x4 matrix of computational nodes into a DSlave. In such a configuration 18 panels of ARUZ are able to simulate a domain of (4 (nodes/FPGA) $\times$ 1 (FPGA/board) $\times$ 12 (boards/panel) $\times$ 18 (panels)) $\times$ (4 (nodes/FPGA) $\times$ 8 (FPGAs/board) $\times$ 12 (boards/panel) $\times$ 1 (panel)), i.e. 864 $\times$ 384 nodes. These estimations have been confirmed by the results of the implementation of the whole DSlave functionality in Vivado (Table II). The number of nodes that can be implemented in a single FPGA is limited by the number of available DSP modules.

TABLE I
SYNTHESIS RESULTS FOR DIFFERENT LBM COLLISION FUNCTIONS FOR THE MATRIX OF 16 LATTICE SITES

| | Bulk | Bottom-Left Corner | Top-Left Corner | Top-Right Corner | Bottom-Right Corner | Left Edge | Right Edge |
|---|---|---|---|---|---|---|---|
| **Clock period [ns]** | 7.63 | 7.64 | 7.64 | 7.64 | 7.64 | 8.30 | 8.30 |
| **Latency [cycles]** | 63 | 43 | 43 | 43 | 43 | 84 | 84 |
| **DSP48E Slices** | 42 | 30 | 30 | 30 | 30 | 27 | 27 |
| **Flip-flops** | 4615 | 3406 | 3406 | 3406 | 3406 | 3752 | 3752 |
| **LUTs** | 4976 | 3454 | 3454 | 3454 | 3454 | 3959 | 3959 |
| **Possible instances** | 17 | 24 | 24 | 24 | 24 | 27 | 27 |

TABLE II
RESOURCE UTILISATION REPORT FOR DSLAVE IMPLEMENTING THE MATRIX OF 16 LATTICE SITES

| Site Type | Used | Available | Util% |
|---|---|---|---|
| **Slice LUTs** | 83789 | 133800 | 62.62 |
| **LUT as Logic** | 83280 | 133800 | 62.24 |
| **LUT as Memory** | 509 | 46200 | 1.10 |
| **LUT as Distributed RAM** | 0 | | |
| **LUT as Shift Register** | 509 | | |
| **Slice Registers** | 77926 | 267600 | 29.12 |
| **Register as Flip Flop** | 77926 | 267600 | 29.12 |
| **Register as Latch** | 0 | 267600 | 0.00 |
| **F7 Muxes** | 254 | 66900 | 0.38 |
| **F8 Muxes** | 66 | 33450 | 0.20 |
| **DSP48E1** | 564 | 740 | 76.22 |

The Vivado HLS clock rate estimations were too pessimistic, as the clock rate of 125 MHz can be achieved. The VHDL simulation of the $2 \times 2$ matrix of DSlaves has indicated, that a single computation cycle takes 1608 ns (see timing diagram in Fig. 5 (a). This result has been confirmed by practical measurements of execution time of $10^9$ LBM cycles in a matrix of $2 \times 2$ DSlaves on a single DBoard (simulation domain: $8 \times 8$ nodes). Extrapolation of these results to the entire ARUZ gives a performance figure of (864 $\times$ 384) / 1608 ns = $206 \cdot 10^3$ MLUPS (Million Lattice Updates per Second, [8]).

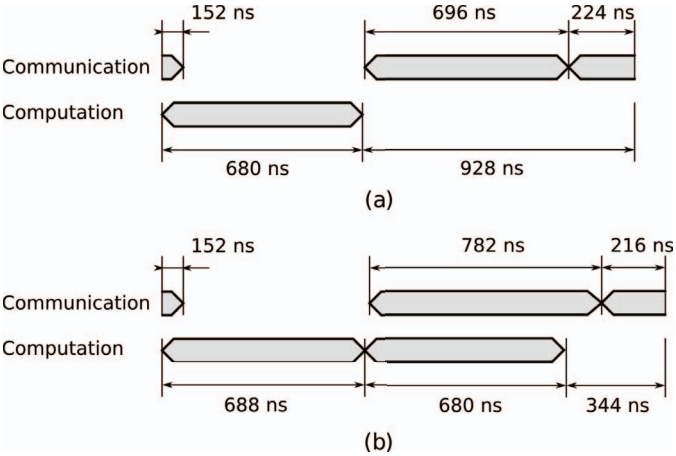



Figure 5. Timing diagram for LBM algorithm implementation: (a) $4 \times 4$ matrix (b) $5 \times 5$ matrix

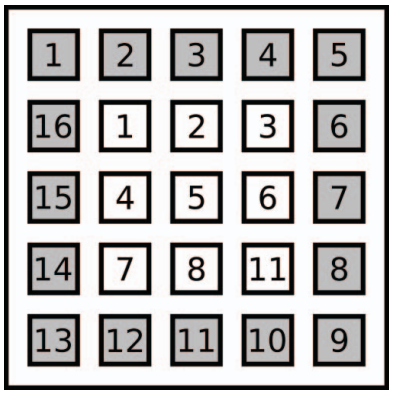



Figure 6. Physical node mapping to lattice sites

The collision phase takes only 42% of the computation cycle. There is a small overlap of computation and communication (152 ns), as the next cycle can be started immediately after reception of the data without waiting for an acknowledgement packet. If a transmission error appears at any of the links, the computation cycle will be suspended after the computation phase until all the data is received correctly and acknowledged by the other party. The computational resources utilization can be improved by increasing the overlap between computation and communication. It can be achieved if a single physical computational node represents more than one lattice site. Such a node will be slightly more complex, as it some cases, apart from performing the computations for domain interior, it would have to perform different computations for the boundary conditions. As it can be seen from the HLS report in Table III, the logic resource utilization increases by ca. 50%, but it is still possible to fit 16 nodes in a single FPGA. The physical synthesis results for this variant are presented in Table III. It is still possible to achieve 8 ns clock cycle. The mapping between physical computational nodes and lattice sites in the part of the domain corresponding to a single FPGA is presented in Fig. 6. The computations for the collision phase in peripheral lattice sites, marked in gray, are performed and their results are transmitted to the neighboring FPGAs (see Fig. 5 (b)). The first transmission sub-phase is ca. 100 ns longer, as three more floating-point values have to be transmitted. Simultaneously with the transmission, 9 of the same physical nodes perform the collision phase computation for the domain interior. The entire computation cycle lasts 1712 ns. As the simulation domain analyzed by ARUZ is now larger ($1080 \times 480$ nodes), the ARUZ performance is increased by 46% to $302 \cdot 10^3$ MLUPS, $76 \cdot 10^3$ times more than the result presented in [9].

TABLE III
SYNTHESIS RESULTS FOR DIFFERENT LBM COLLISION FUNCTIONS FOR THE MATRIX OF 25 LATTICE SITES

| | Bulk | Bulk & Bottom-Left | Bulk & Top-Left | Bulk & Top-Right | Bulk & Bottom-Right | Bulk & Left | Bulk & Right |
|---|---|---|---|---|---|---|---|
| **Clock period [ns]** | 7.63 | 7.64 | 7.50 | 7.50 | 7.50 | 8.30 | 8.30 |
| **Latency [cycles]** | 63 | 64 | 64 | 64 | 64 | 85 | 85 |
| **DSP48E Slices** | 42 | 42 | 42 | 42 | 42 | 42 | 42 |
| **Flip-flops** | 4615 | 5065 | 5065 | 5065 | 5065 | 4958 | 4958 |
| **LUTs** | 4976 | 5413 | 5394 | 5388 | 5392 | 5544 | 5547 |
| **Possible instances** | 17 | 17 | 17 | 17 | 17 | 17 | 17 |

TABLE IV
RESOURCE UTILISATION REPORT FOR DSLAVE IMPLEMENTING THE MATRIX OF 25 LATTICE SITES

| Site Type | Used | Available | Util% |
|---|---|---|---|
| **Slice LUTs** | 84338 | 133800 | 63.03 |
| **LUT as Logic** | 83831 | 133800 | 62.65 |
| **LUT as Memory** | 507 | 46200 | 1.10 |
| **LUT as Distributed RAM** | 0 | | |
| **LUT as Shift Register** | 507 | | |
| **Slice Registers** | 80380 | 267600 | 30.21 |
| **Register as Flip Flop** | 80380 | 267600 | 30.21 |
| **Register as Latch** | 0 | 267600 | 0.00 |
| **F7 Muxes** | 280 | 66900 | 0.42 |
| **F8 Muxes** | 46 | 33450 | 0.42 |
| **DSP48E1** | 534 | 740 | 72.16 |

## V. COMPARISON WITH SUPERCOMPUTER IMPLEMENTATION

The LBM algorithm scales perfectly on ARUZ due to the possibility of overlapping of communication and computation. The same principle can be applied to obtain a good scalability on a supercomputer, however larger subdomains need to be allocated to each of the nodes. Figure 7 presents the performance

obtained on the Prometheus [10] cluster in Cracow vs. the number of cores allocated for the computations. Each physical core has been computing the results for the subdomain of 1000 $\times$ 1000 lattice nodes.

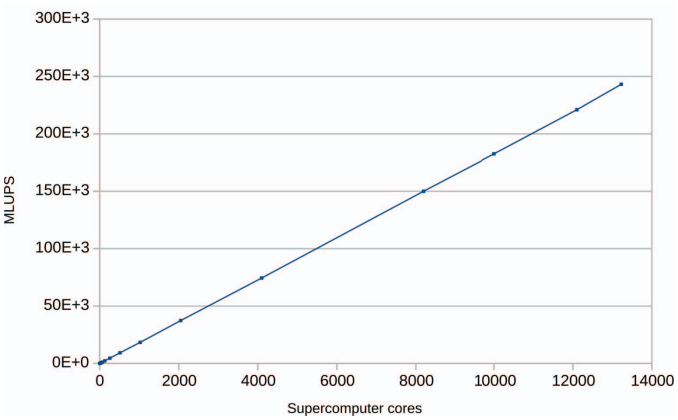



Figure 7. Performance results for simulation on the Prometheus supercomputer.

The batch scheduling system has allowed allocation of up to 13225 cores for our tests. In such a case we were able to achieve $243 \cdot 10^3$ MLUPS. Linear extrapolation of this result to the entire machine, containing 55728 cores, gives $1024 \cdot 10^3$ MLUPS. ARUZ and Prometheus consume ca. 100 kW and 800 kW, respectively. This results in the power efficiency of 3000 MLUPS/kW for ARUZ and 1280 MLUPS/kW for Prometheus.

## VI. Conclusion

The proposed improvement to the Lattice Boltzmann method implementation on ARUZ increases the performance by 46%, up to $302 \cdot 10^3$ MLUPS.

In theory, there is still a room for improvements, as the communication phase does not fully overlap the computation phase and the number of DSP blocks would allow to fit 17 computational blocks into the FPGA. Therefore, the maximum achievable performance improvement is $1608/680 \times 17/16 = 2.51$. The theoretical performance improvement assuming linear transmission time increase in sub-phase 1 for different grid sizes is presented in Table V. The transmission can be performed in two sub-phases, the first one sending the data from 4 nodes on each border ($4 \times 3 \times 32$ bits, 728 ns), the second one combining the data for diagonal transmission obtained in the first sub-phase and the data from the remaining boundary nodes. For N above 5, the communication latency can be completely hidden, the maximum speedup is achieved for N=17. However, increasing N would introduce additional data multiplexers and increase the clock cycle time. In addition, the current implementation already exhibits routing congestions that would get even worse in such a case.

The presented approach can be applied not only to the Lattice Boltzmann method, but to any algorithm that needs only a local communication between neighboring nodes.

TABLE V
Estimated performance improvement for different grid sizes

| N | Transmission time [ns] | Total phases | Peripherial phases | Cycle time [ns] | Performance improvement |
|---|---|---|---|---|---|
| 5 | 1016 | 2 | 1 | 1704 | 1.47 |
| 6 | 1112 | 3 | 2 | 2048 | 1.77 |
| 7 | 1208 | 3 | 2 | 2048 | 2.40 |
| 8 | 1304 | 4 | 2 | 2728 | 2.36 |
| 9 | 1400 | 5 | 2 | 3408 | 2.39 |
| 10 | 1496 | 6 | 3 | 4088 | 2.46 |
| 11 | 1592 | 8 | 3 | 5448 | 2.23 |
| 12 | 1688 | 9 | 3 | 6128 | 2.36 |
| 13 | 1784 | 10 | 3 | 6808 | 2.49 |
| 14 | 1880 | 12 | 4 | 8168 | 2.41 |
| 15 | 1976 | 14 | 4 | 9528 | 2.37 |
| 16 | 2072 | 16 | 4 | 10888 | 2.36 |
| 17 | 2168 | 17 | 4 | 11568 | 2.51 |

## References

[1] T. Pakuła and J. Teichmann, "Model for relaxation in supercooled liquids and polymer melts," in *Materials Research Society Symposium – Proceedings, Volume 455*, 1996, p. 211.

[2] P. Polanowski, J. Jung, and R. Kielbik, "Special purpose parallel computer for modelling supramolecular systems based on the dynamic lattice liquid model," *Computational Methods in Science and Technology*, vol. 16, no. 2, pp. 147–153, 2010.

[3] K. Hałagan, P. Polanowski, J. Jung, and M. Kozanecki, "Modelling of complex liquids with cooperative dynamics using ARUZ," in *Dedicated parallel machines – a breakthrough in computation ARUZ-Workshop 2016, Lodz, Poland, 1-3 December 2016*, 2016, pp. 10–11.

[4] R. Kiełbik, K. Hałagan, W. Zatorski, J. Jung, J. Ulański, A. Napieralski, K. Rudnicki, P. Amrozik, G. Jabłoński, D. Stożek, P. Polanowski, Z. Mudza, J. Kupis, and P. Panek, "ARUZ — large-scale, massively parallel FPGA-based analyzer of real complex systems," *Computer Physics Communications*, pp. –, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465518302182

[5] G. R. McNamara and G. Zanetti, "Use of the Boltzmann equation to simulate lattice-gas automata," *Phys. Rev. Lett.*, vol. 61, pp. 2332–2335, Nov 1988.

[6] G. Jabloński and J. Kupis, "The application of high level synthesis for implementation of lattice Boltzmann method in ARUZ," *International Journal of Microelectronics and Computer Science*, vol. 8, no. 1, pp. 36–42, 2017.

[7] U. Frisch, B. Hasslacher, and Y. Pomeau, "Lattice-gas automata for the Navier-Stokes equation," *Phys. Rev. Lett.*, vol. 56, pp. 1505–1508, Apr 1986.

[8] A. G. Shet, S. H. Sorathiya, S. Krithivasan, A. M. Deshpande, B. Kaul, S. D. Sherlekar, and S. Ansumali, "Data structure and movement for lattice-based simulations," *Phys. Rev. E*, vol. 88, p. 013314, Jul 2013.

[9] K. Sano, O. Pell, W. Luk, and S. Yamamoto, "FPGA-based streaming computation for lattice Boltzmann method," in *2007 International Conference on Field-Programmable Technology, ICFPT 2007, Kitakyushu, Japan, December 12-14, 2007*, 2007, pp. 233–236.

[10] "Prometheus on top500 list," 2017. [Online]. Available: https://www.top500.org/system/178534

**Grzegorz Jabłoński** was born in 1970. He received MSc and PhD degrees in electrical engineering from Lodz University of Technology in 1994 and 1999 respectively. He is currently an Assistant Professor in the Department of Microelectronics and Computer Science Lodz University of Technology. His research interests include compiler construction, microelectronics, simulation of electronic circuits and semiconductor devices, thermal problems in electronics, digital electronics, embedded systems and programmable devices.

**Joanna Kupis** is a PhD student in the Department of Microelectronics and Computer Science, Lodz University of Technology.