Tikrit Journal of Engineering Sciences (2025); 32(Sp1): 2823
DOI: http://doi.org/10.25130/tjes.sp1.2025.44

IRAQI Academic Scientific Journals

ISSN: 1813-162X (Print); 2312-7589 (Online)
Tikrit Journal of Engineering Sciences
available online at: http://www.tj-es.com

TJES
Tikrit Journal of Engineering Sciences

A Specialised Computing Device for Multiplying Square Binary Matrices Based on Multiport Parallel-Pipelined Memory

Mohammed H. Najajra *[a], **Aleksei V. Bolgak** [b], **Eduard I. Vatutin** [b], **Ismail F. Amer** [c]

a Al-Istiqlal University, Jericho, Palestine.
b Southwest State University, Kursk, Russia.
c Kazan Federal University, Kazan, Russia.

Keywords:
Algorithmic optimization; Binary matrix; Matrix multiplication; Multiport memory; Parallel-pipeline memory; Systolic device; Transitive closure.

Highlights:

- *Classical and buffered multiplication of real matrices.*
- *Performance of block multiplication of real matrices depending on the block size.*
- *Estimation of the time spent by computing devices on matrix multiplication.*

ARTICLE INFO

Article history:

Received	16 Nov. 2025
Received in revised form	28 Dec. 2025
Accepted	04 Jan. 2026
Final Proofreading	05 Jan. 2026
Available online	05 Jan. 2026

Citation: Najajra MH, Bolgak AV, Vatutin EI, Amer IF. **A Specialised Computing Device for Multiplying Square Binary Matrices Based on Multiport Parallel-Pipelined Memory**. *Tikrit Journal of Engineering Sciences* 2025; **32**(Sp1): 2823.
http://doi.org/10.25130/tjes.sp1.2025.44

*Corresponding author:
Mohammed H. Najajra
Al-Istiqlal University, Jericho, Palestine.

Abstract: This article considers matrix multiplication in the problem of finding the transitive closure of a binary relation with the transitivity property, as well as in the construction of the reachability and counter-reachability matrices in general graphs. An analysis of approaches to practical implementation for finding the transitive closure of a binary relation is presented: the Floyd-Warshall algorithm and raising the adjacency matrix to a power until it stabilises. The problem of processing large (thousands to millions of elements) graph diagrams of parallel algorithms on a processor (CPU), and the primary methods for optimising matrix calculations at both the software (algorithmic) and hardware levels, are considered. The main types of digital devices based on the parallel-pipeline data-processing principle are identified, and their advantages and disadvantages are outlined. A specialised computing device for fast multiplication of square binary matrices of size n × n is considered, whose distinctive feature is pipelining the data read operation from a specialised multiport memory. A mathematical model and a method for organising the parallel-pipeline memory of a specialised square binary matrix multiplication device are presented. An estimate of the matrix-processing time and hardware complexity for the developed and prototype devices is presented. Computational experiments showed that, despite a slightly higher hardware complexity (up to 8.8×) than the prototype device, the proposed device multiplies square binary matrices of size n ≤ 512 up to 52.4× faster. This represents a significant advantage when implemented in a semi-custom design using field-programmable gate arrays or a custom design based on application-specific integrated circuits. In this paper, we present a novel systolic device whose core innovation is a pipelined multiport memory architecture. By ensuring a continuous, high-bandwidth data flow to the processing elements, our contribution enables the systolic array to operate at its theoretical peak performance.

1.INTRODUCTION

Many computationally intensive tasks involve matrix multiplication. The effectiveness of its implementation determines how quickly these tasks are solved. The applications of high-performance computing based on matrix computing include: real-time systems that make decisions in a short time interval (no more than a few tens of milliseconds), computer-aided design (CAD) systems [1], satellite and inertial navigation systems, UAV swarm control algorithms with cluster analysis elements [2], and graph problems with building routes and using adjacency matrices in graphs [3]. When solving many issues in separate mathematics, it is necessary to multiply binary matrices. These include constructing a reachability-counter-reachability matrix in general graph-theoretic graphs [4] and computing the transitive closure of a binary relation [5]. There are two approaches to the practical implementation of the transitive closure search. The first one is based on the Floyd-Warshall algorithm [6, 7]. This algorithm implements a special order of consideration of matrix elements, which allows you to find the reachability matrix in one pass, which is a key advantage of this algorithm, but, along with this, there is a disadvantage because the Floyd-Warshall algorithm cannot be parallelised, as it depends on the order in which multiplication operations are performed. The temporal asymptotics of the Floyd-Warshall algorithm are $O(n^3)$. The second approach is based on squaring the adjacency matrix until it stops changing. The resulting value of a matrix with the property of transitive closure will be obtained in the worst case in $\lceil \log_2 n \rceil$steps (squaring the matrix), and the time asymptotic value of the algorithm will be $O(n^3 \log n)$. From this, we conclude that the time complexity of this approach is higher than that of the Floyd-Warshall algorithm; however, it allows effective parallelisation because it does not depend on the order of operations. Based on this analysis of two algorithmic approaches, the subsequent methodology will involve a comparative evaluation to determine the optimal implementation strategy for different practical scenarios. This evaluation will be conducted by theoretically assessing both time and space complexity and by implementing algorithms to test their performance on a set of sample relations of varying sizes and densities. The parallelizable nature of the matrix squaring approach will also be explored by benchmarking its performance in a multi-threaded computing environment against the serial execution of the Floyd-Warshall algorithm.

2.METHODOLOGY

Processing large graph circuits (thousands to millions of elements) for parallel algorithms on modern CPUs can take several minutes to several hours. In this context, software and hardware approaches are employed to optimise matrix calculations. The software implementation employs a basic approach to matrix multiplication: classical multiplication. This approach is ineffective when the matrices exceed the CPU cache size. Based on this, various algorithmic methods are used in practice to reduce CPU cache misses and improve overall system performance. For example, matrix column-buffered multiplication (Fig. 1) or block multiplication (Fig. 2) enables efficient use of the CPU cache [8]. Figure 1 demonstrates that buffered multiplication significantly outperforms the classical method for large matrices by reducing cache misses.

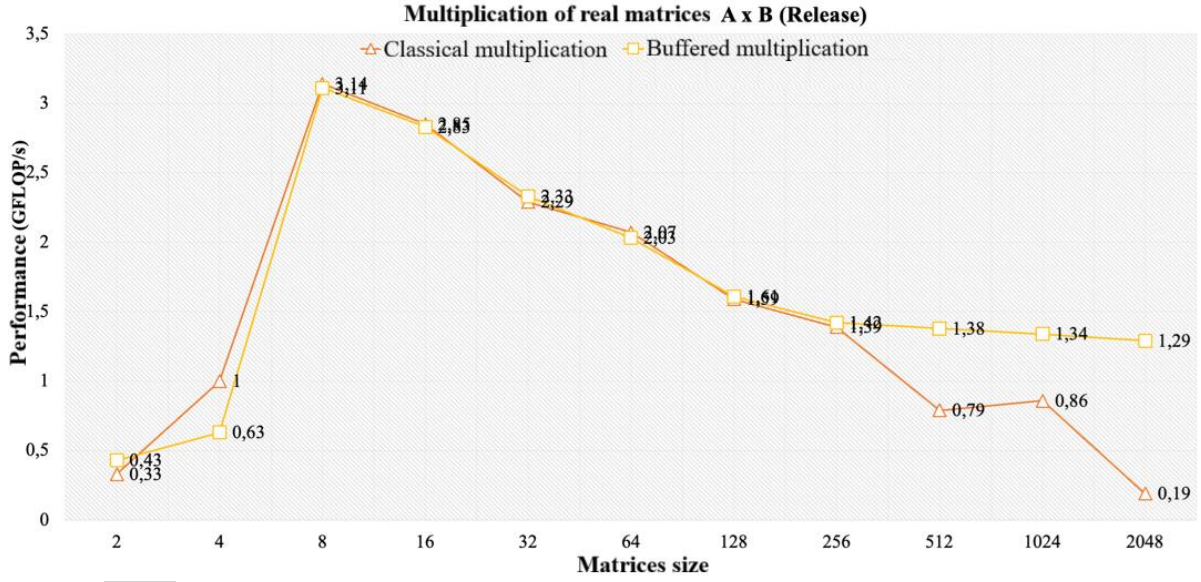

Fig. 1 Performance Graphs of Classical and Buffered Multiplication of Real Matrices.

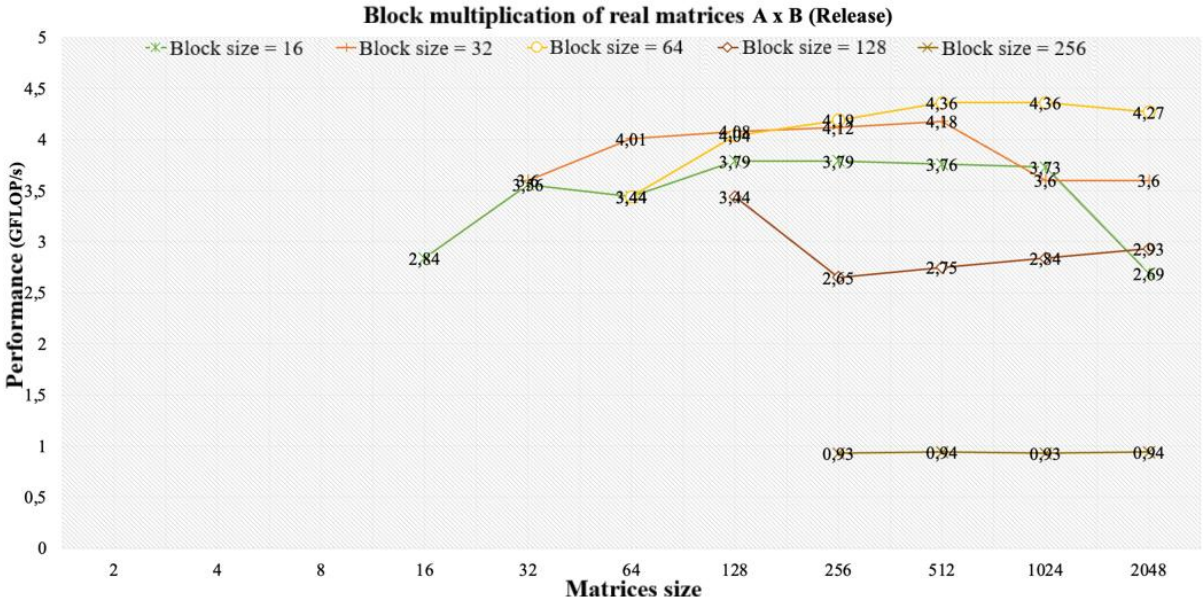

Fig. 2 Graphs of the Performance of Block Multiplication of Real Matrices Depending on the Block Size.

Another well-known approach to reduce the time spent on matrix computations is to perform matrix multiplication on graphics processing units (GPUs) with shared memory (GPGPUs). The use of parallel software implementations, such as CUDA, OpenCL, and STREAM, for GPU computing [9-11] also enables higher system performance. If, at the software level, the execution time of the matrix multiplication operation is unacceptably long, then it is justified to transfer this operation to the hardware level. Approaches to implementing matrix operations at the hardware level are divided into three main groups, as presented in Table 1.

Table 1 Types of Matrix Processing Devices.

Group	Description
Devices based on optical elements [12-14]	Devices in this group are not currently used in practice.
Devices based on analogue probabilistic principles of signal processing [15]	Noncompliance with IEEE 754 standards and low computational accuracy in this group limit their use in computing.
Digital devices based on parallel pipeline architecture [16-18]	The devices in this group are based on parallel and pipelined data processing. On these devices, matrix multiplication is performed in linear time, yielding significant performance gains.

There is a separate class of tasks aimed at processing binary matrices. These include, for example, the above-described functions for constructing reachability and counter-reachability matrices in graphs and for computing the transitive closure of binary relations with the transitivity property. In their hardware implementation, it is possible to significantly reduce hardware complexity and improve the performance of specialised computing devices. These devices can be classified into two categories: systolic and iterative. Computing devices with a systolic structure are characterised by high performance, ease of implementation due to their regularity, and ease of reconfiguration; however, they exhibit significantly greater hardware complexity, which is an obstacle to their practical implementation when multiplying large matrices. Iterative binary matrix processing devices, focused on the hardware implementation of classical multiplication algorithms, are characterised by moderate speed and low hardware complexity. In each specific case (matrix size, matrix density, and hardware-complexity limitations), one of the above software or hardware approaches can be selected to implement matrix multiplication in practice.

3.RESULTS

Based on the above information, this article proposes a device that implements the systolic multiplication principle to reduce the time spent reading data using a specialised multi-port memory. In the prototype device [19], a corresponding structural and functional organisation of a multi-port memory was proposed, enabling the reading of 2n pairs of matrix coefficients per clock cycle, significantly outperforming classical memories (DDR or GDDR), which read only one operand per clock cycle. During the performance evaluation [20], it was found that for matrices with $n \geq 64$, the device's operating time (conveyor cycle) is

limited by the rate of data transfer from memory. Based on the performed analysis, to reduce the time spent on reading data from multiport memory, a device is proposed (see Figs. 3 and 4) based on the systolic multiplication principle, a distinctive feature of which is pipelining data reading [21]. Figure 3 shows a functional diagram of the proposed device, which includes $n \times n$ cells of operating blocks (OUC), where n is the size of the square matrices being multiplied, blocks of matrix coefficients (BMC) 2, 3, a shift register 4, and a group of n two-stage registers 5. Figure 4 shows the cell diagram of the device's operating unit, consisting of two-stage triggers 6, 7, 8; logic gates OR 10, 13; logic gates AND 9, 11, 12; and an inverter 14. Figure 5 shows a diagram of a single memory-cell storage unit (specialised pipelined multiport memory) consisting of trigger 16, logic element *AND* 17, a group of n elements *AND* 18, a group of n elements *OR* 19, and blocks of n two-stage triggers 28, 29, 30. Figure 6 shows a diagram of a matrix coefficient block (one cell for storing one matrix coefficient within a multiport memory), which contains $n \times n$ storage blocks 15, two groups of $(n\ 1)$ two-stage registers 39, 40, a group of $n \times n$ two-stage triggers 41, and a group of $n \times (n\ 1)$ logic gates *OR* 42. Here is a description of a mathematical model for organising the operation of a specialised multiport conveyor memory (Fig. 6), comprising n × n storage units (Fig. 5) [22]. The proposed specialised device memory operates according to the following mathematical model:

$$39_i^{(t)} := Rg\ 39_{i-1}^{(t-1)},\ i = \overline{2, n}, \quad (1)$$

$$40_j^{(t)} := Rg\ 40_{j-1}^{(t-1)},\ j = \overline{2, n}, \quad (2)$$

$$29_k^{(t)}[i, j] := TT\ 29_k^{(t-1)}[i, j-1],\ i = \overline{1, n},\ j = \overline{2, n},\ k = \overline{1, n}, \quad (3)$$

$$28_k^{(t)}[i, j] := TT\ 28_k^{(t-1)}[i-1, j],\ i = \overline{2, n},\ j = \overline{1, n},\ k = \overline{1, n}, \quad (4)$$

$$TT\ 30_k^{(t)}[i, j] := D[i, j]\ \&\ TT\ 29_k^{(t)}[i, j]\ \&\ TT\ 28_k^{(t)}[i, j] \vee TT\ 30_k^{(t-1)}[i-1, j],\ i = \overline{2, n},\ j = \overline{1, n},\ k = \overline{1, n}, \quad (5)$$

$$TT\ 41^{(t)}[k, 1] := TT\ 30_k^{(t-1)}[n, 1],\ k = \overline{1, n},\ 41^{(t)}[k, j] := TT\ 41^{(t-1)}[k, j-1] \vee TT\ 30_k^{(t-1)}[n, j],\ j = \overline{2, n},\ k = \overline{1, n}. \quad (6)$$

Here, i, $j = \overline{1, n}$ Are the numbers of the current storage unit, $k = \overline{1, n}$ Is the read port number, and t is the clock cycle number. Formula (1) corresponds to the conveyor principle of advancing addresses by columns, and formula (2) corresponds to the conveyor principle of advancing addresses by rows (addresses are specified in a unitary code of the form 00... 01000...00, where one indicates the row and/or column number of the selected storage unit). Formulas (3) and (4) describe the pipelining of these addresses through the storage units of memory cells, and formula (5) describes the reading and subsequent pipelining of data from the selected storage unit down the columns. Formula (6) describes the further pipelining of the read data to the output of the matrix coefficient block. During software processing, the matrix is stored in RAM as a two-dimensional array of binary values (0 or 1). The matrix elements are stored in memory in row order, and address calculations are required to access them. In hardware-oriented processing, data are transferred from RAM to a specialised multiport, parallel-pipelined memory within a computing device, which can partition a model in space by stage. Next, the matrices are fed into the systolic computing device, one top-down and the other left-to-right, where they are multiplied in linear time. During the assessment of the time spent processing matrices on the specialised computing device developed, the following results were obtained [23] and are presented in Tables 2–6. Figures 7 and 8 present time diagrams that explain the logic of the developed device. The time-cost values for each work stage of the developed specialised computing device and the prototype device, calculated for various n values with t_0 = 1 *ns*, are shown in Table 2. Based on the data in Table 2, the loading and unloading times for the source and result data of the proposed device are significantly higher than the matrix multiplication time, making frequent matrix loading and unloading impractical. For example, when performing the transitive closure of a binary relation represented as a binary matrix, the initial matrix is loaded once, followed by a series of squarings, which is effectively implemented by the proposed device. A comparative estimate of the time spent operating the developed specialised computing device and prototype, computed for various n values with t_0 = 1 *ns*, is shown in Table 3. A time diagram illustrating the operational stages of the developed specialised computing device is shown in Fig. 7.

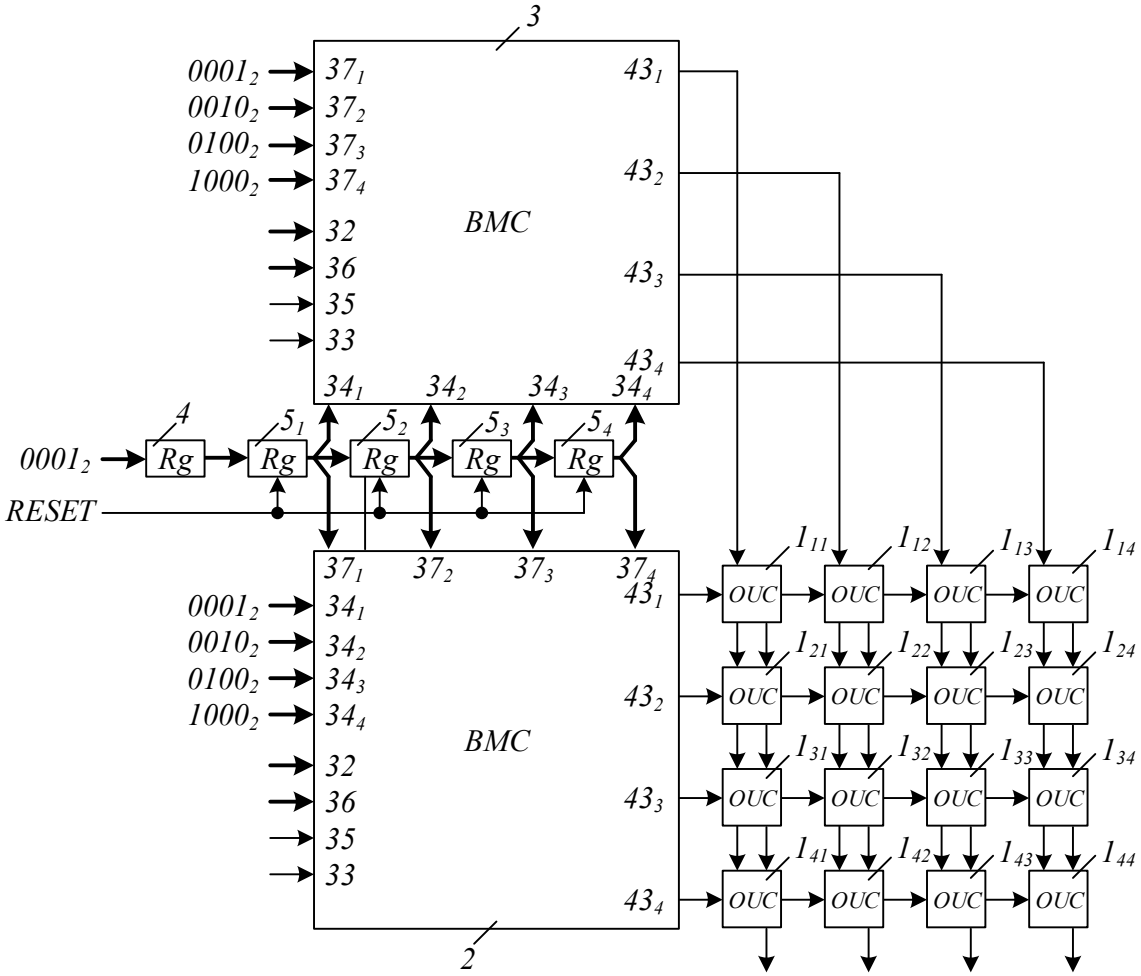

Fig. 3 Structural and Functional Diagram of a Specialised Computing Device for Multiplying Square Binary Matrices.

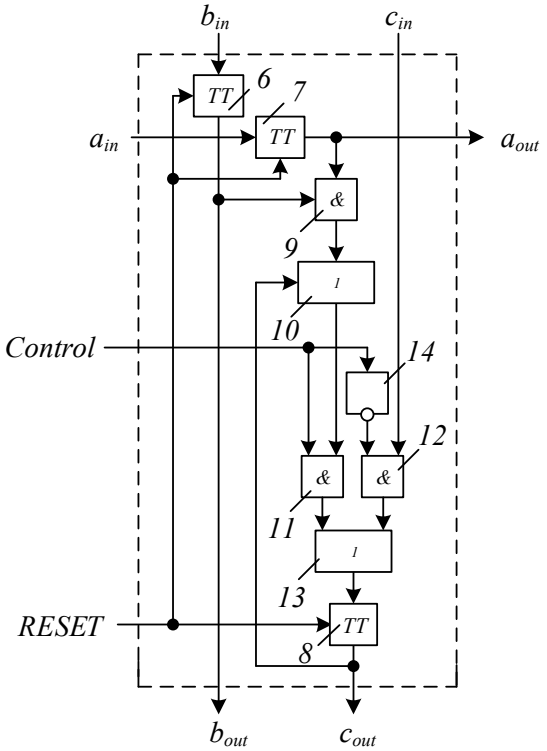

Fig. 4 The Cell Diagram of the Operational Unit of the Computing Device for Multiplying Binary Matrices.

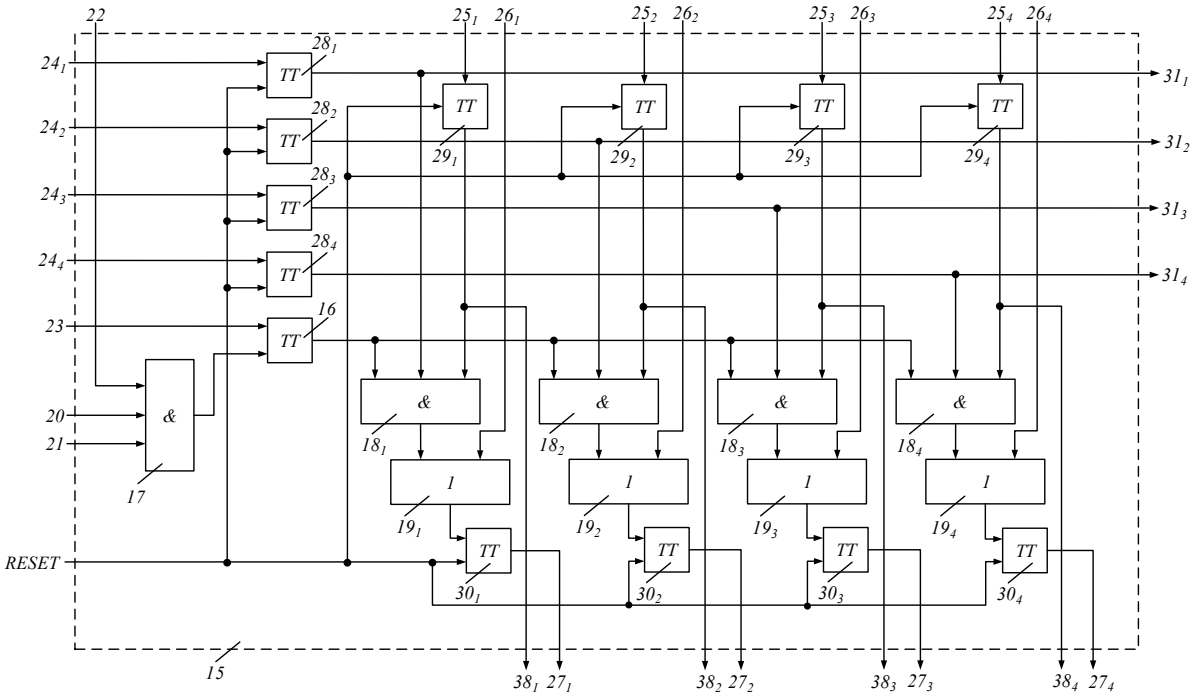

Fig. 5 Cell Diagram of the Storage Unit of the Computing Device for Multiplying Binary Matrices.

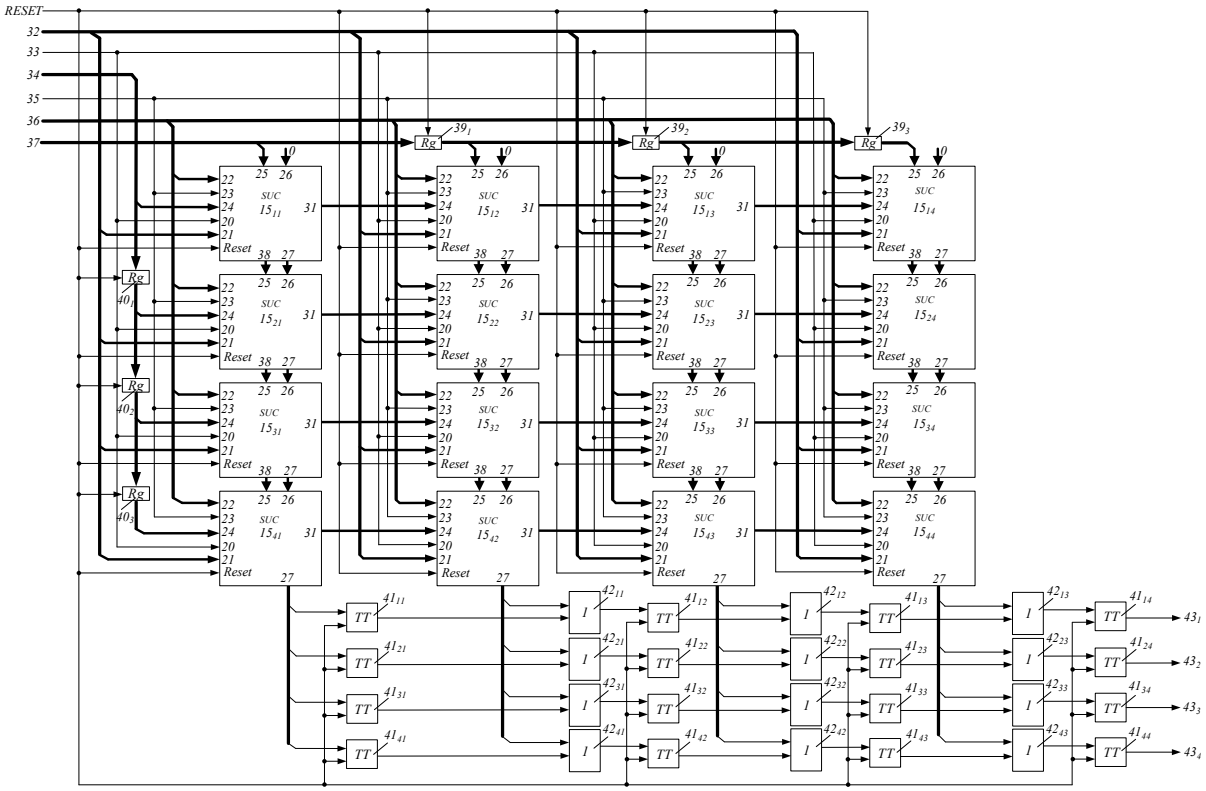

Fig. 6 Diagram of the Matrix Coefficient Block of the Binary Matrix Multiplication Computing Device.

Table 2 Time Cost Values for Each Stage of the Device Operation.

n	$writing, ms$	$writing_{prototype}, ms.$	$working, Ms$	$working_{of\ prototype}, ms$	$t_{winning}, ms$	$t_{winning\ prototype}, ms$	$t_{gen.\ prototype}, ms$	$t_{gen.}, ms$
2	0.000032		0.00003	0.000015	0.000024		0.000073	0.000088
4	0.000128		0.00007	0.000056	0.000096		0.000282	0.000296
8	0.000512		0.00015	0.000195	0.000384		0.00109	0.00104
16	0.00205		0.00031	0.00068	0.00154		0.00427	0.00389
32	0.00819		0.00063	0.00245	0.00614		0.0168	0.0149
64	0.0327		0.00127	0.00914	0.0246		0.0664	0.0586
128	0.131		0.00255	0.0349	0.0983		0.264	0.232
256	0.524		0.00511	0.135	0.393		1.05	0.922
512	2.09		0.0102	0.535	1.57		4.21	3.68
1024	8.39		0.0204	2.12	6.29		16.8	14.7
2048	33.5		0.0409	8.44	25.2		67.1	58.7

Table 3 Comparative Assessment of the Time Spent on the Operation of the Devices.

n	$T_{gen.\ prototype}, ms$	t_{gen}, ms	Winning *Times*	%
8	0.00109	0,00104	1.04	4
16	0.00427	0.00389	1.10	10
32	0.0168	0.0149	1.12	12
64	0.0664	0.0586	1.13	13
128	0.264	0.232	1.14	14
256	1.05	0.922	1.14	14
512	4.21	3.68	1.14	14
1024	16.8	14.7	1.14	14
2048	67.1	58.7	1.14	14

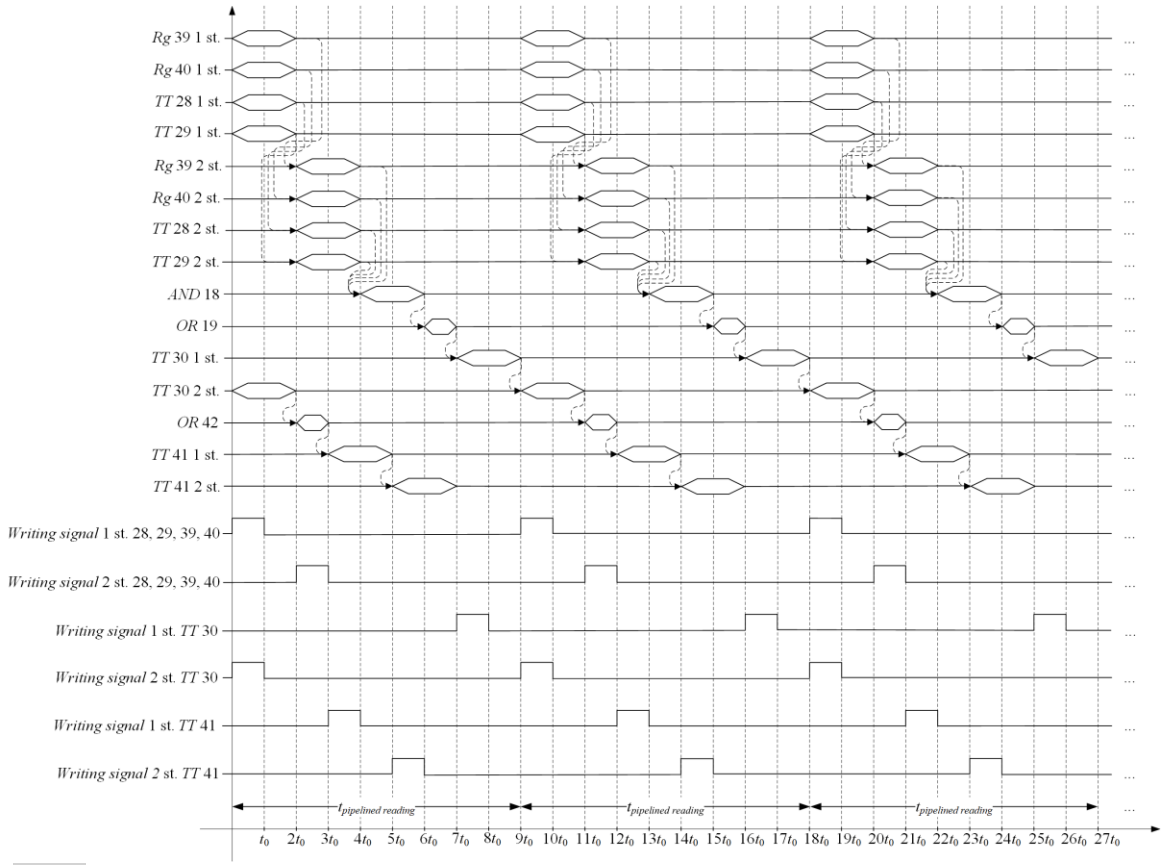

Fig. 7 Time Diagram Explaining the Stage of Operation of the Developed Specialised Computing Device.

A time diagram illustrating the operational stages of the developed specialised computing device's operating unit (see Fig. 4) is shown in Fig. 8. A comparative estimate of the time spent on the operation of the developed specialised computing device for multiplying square binary matrices and a prototype in the task of searching for a transitive closure of a binary relation (the multiplication operation is performed $\log_2 n$Es), calculated for various n, $t_0 = 1\ ns$, is shown in Table 4. The time-cost values for the matrix multiplication operation on the developed specialised computing device and the prototype device, computed for various n and $t_0 = 1\ ns$, are shown in Table 5. An estimate of the time spent by known computing devices on matrix multiplication, computed for various n values and $t_0 = 1$, is shown in Table 6. Based

on the results of the computational experiment, it can be concluded that using pipelining to read data from a specialised multiport memory reduces the processing time for square binary matrices with n ≤ 2048 by a factor of 206.3. The hardware complexity of the developed specialised computing device for multiplying square binary matrices, with pipelining of the data read operation from a specialised multiport memory, is estimated for a prototype device at the same n [30]. The hardware complexity was estimated in equivalent valves (EVs), where an EV is a one- or two-input logic element performing an elementary logical operation. The hardware-complexity values for the proposed square binary matrix multiplication device with pipelined data read from a specialised multiport memory are shown in Table 7 for various n values. The hardware complexity values for the specialised memory, the systolic part of the prototype, and the developed specialised computing device, computed for various n values, are shown in Table 8. The estimates of hardware complexity (see Table 8) indicate that most of the equivalent gates are allocated to implementing specialised memory. The hardware complexity values for the proposed device and prototype, computed for various n values, are shown in Table 9. From the data presented (see Tab. 9), it follows that the prototype device has 5.5–8 times lower hardware complexity than the developed specialised matrix multiplication computing device with pipelined data reading from a specialised multiport memory, depending on the matrix size *n*. Based on an estimate of the time cost and hardware complexity of the developed device (see Fig. 9), it can be concluded that it is the most productive among known devices when multiplying matrices of size 8 × ≤ 512. The lower boundary on the graph (Fig. 9) indicates that the developed computing device is not advisable for multiplying square binary matrices of size n × 8, given its relatively high hardware complexity for such a small matrix size. The upper bound indicates that the developed computing device will most likely not fit within a modern FPGA crystal when multiplying matrices of size $n \geq 512$. Practical implementation of the developed specialised computing device using an FPGA (Figs. 10 and 11) [31, 32].

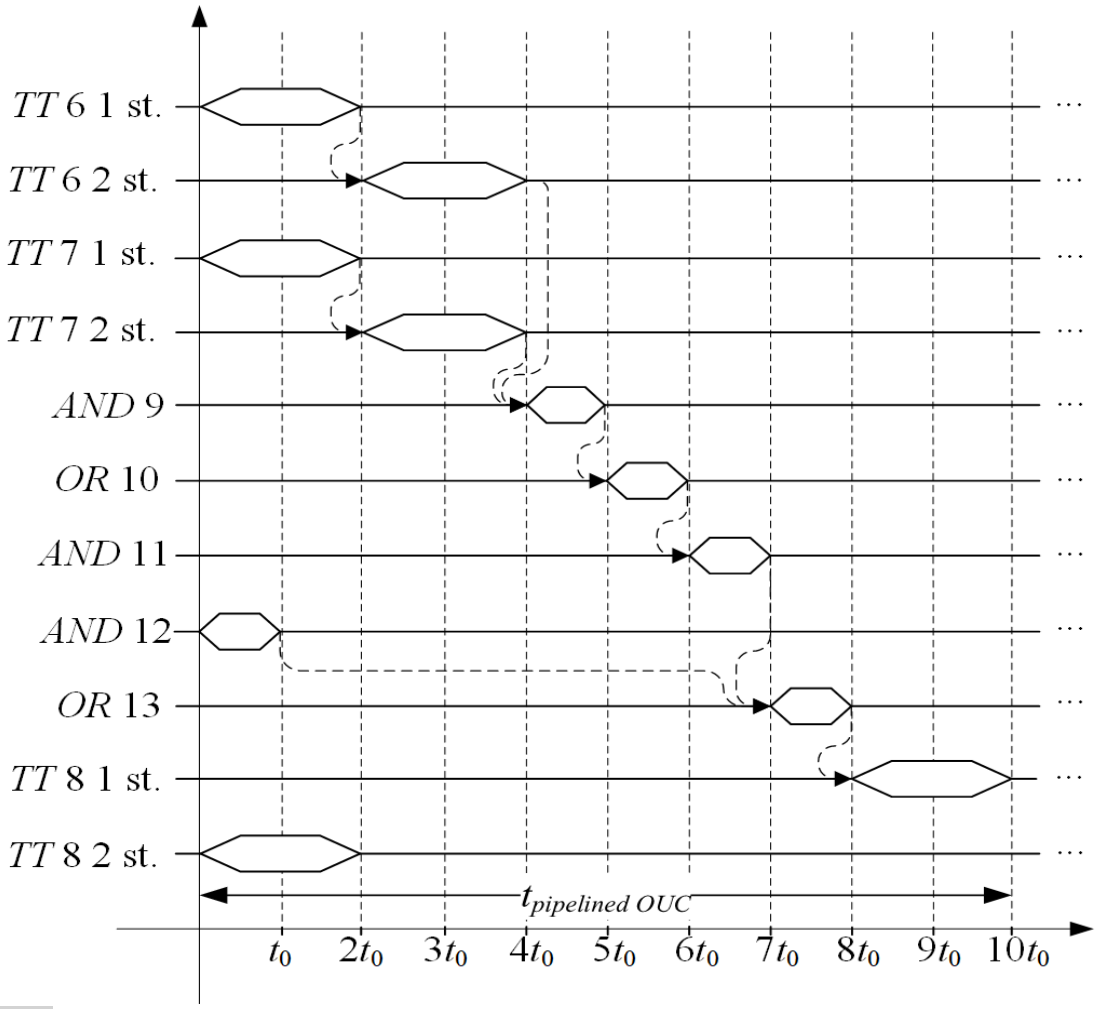

Fig. 8 Time Diagram Explaining the Operation of the Operating Unit of the Developed Specialised Computing Device.

Table 4 Comparative Assessment of the Time Spent on the Operation of Devices.

n	$t_{gen.\ prototype}$, *ms*	$t_{gen.}$, *ms*	Winning, *times*
8	0.00148	0.00134	1.10
16	0.00631	0.00482	1.31
32	0.0266	0.0174	1.52
64	0.112	0.0649	1.73
128	0.473	0.247	1.92
256	2.00	0.958	2.09
512	8.48	3.76	2.25
1024	35.9	14.8	2.41
2048	151	59.1	2.56

Table 5 Comparative Estimation of the Time Spent by Devices on the Matrix Multiplication Operation.

n	The developed device (favourable decision on the grant of an RF patent for an invention, application No. 2025104287)	Prototype (RF patent for utility model No. 193927)	Winning, *times*
8	150 *as*	190 *s*	1.3
16	310 *as*	680 *as*	2.2
32	630 *as*	2450 *is*	3.9
64	1.27 *ms*	9.14 *ms*	7.2
128	2.55 *ms*	34.9 *ms*	13.7
256	5.11 *ms*	135 *ms*	26.4
512	0.0102 *ms*	0.535 *ms*	52.4
1024	0.0204 *ms*	2.12 *ms*	103.9
2048	0.0409 *ms*	8.44 *ms*	206.3

Table 6 Estimation of the Time Spent by Computing Devices on Matrix Multiplication.

n	Intel Core i5-11400 [5]	Intel Core i7-4770 [24]	Intel Core i7-4770 (*SSE*) [25, 26]	NVIDIA GeForce 770 GTX [27]	Analogue (RF patent No. 157948) [28]	Analogue (RF patent No. 2744239) [29]	Prototype (RF patent for utility model No. 193927) [19]	The developed device (favourable decision on the grant of an RF patent for an invention, application No. 2025104287) [21]
8	330 *as*	–	–	–	145000 *as*	3900 *is*	190 *s*	150 *as*
16	2870 *is*	–	1320 *is*	–	145000 *as*	24300 *is*	680 *as*	310 *as*
32	28200 *is*	–	9630 *is*	–	145000 *as*	175000 *is*	2450 *is*	630 *as*
64	300 *ms*	–	58.2 *ms*	–	198 *ms*	1310 *ms*	9.14 *ms*	1.27 *ms*
128	2600 *mas*	–	419 *ms*	–	391 *mas*	10100 *ms*	34.9 *ms*	2.55 *ms*
256	23600 *ms*	14000 *ms*	3078 *ms*	55000 *ms*	776 *ms*	81400 *ms*	135 *ms*	5.11 *ms*
512	194 *ms*	104 *ms*	22,7 *ms*	75 *ms*	1.54 *ms*	656 *ms*	0.535 *ms*	0.0102 *ms*
1024	1600 *ms*	850 *ms*	239 *ms*	210 *ms*	3.08 *ms*	4890 *ms*	2.12 *ms*	0.0204 *ms*
2048	13300 *ms*	7000 *ms*	2450 *ms*	1300 *ms*	6.15 *ms*	39100 *ms*	8.44 *ms*	0.0409 *ms*

Table 7 Evaluation of the Hardware Complexity of the Developed Specialised Computing Device.

n	The developed device is *an EV*.
10	6.1×10^4
100	5.4×10^7
1000	5.4×10^{10}

Table 8 Evaluation of the Hardware Complexity of the Specialised Memory and the Systolic Part of the Prototype Device, and the Developed Specialised Computing Device.

n	$R_{RAM\ prototype}$, *EV*	R_{RAM}, *EV*	$R_{syst.\ prototype}$, *EV*	$R_{syst.}$, *EV*
2	140	588	120	
4	744	4072	480	
8	4496	30096	1920	
16	30240	230944	7680	
32	2.2×10^5	1.8×10^6	3.0×10^4	
64	1.7×10^6	1.4×10^7	1.2×10^5	
128	1.3×10^7	1.1×10^8	4.9×10^5	
256	1.0×10^8	9.1×10^8	2.0×10^6	
512	8.1×10^8	7.2×10^9	7.9×10^6	
1024	6.5×10^9	5.8×10^{10}	3.1×10^7	
2048	5.2×10^{10}	4.6×10^{11}	1.2×10^8	

Table 9 Comparative Assessment of the Hardware Complexity of the Developed Specialised Computing Device with the Prototype Device.

n	The developed device (a favourable decision to grant an RF patent for an invention or an application). No. 2025104287), *EV*	Prototype (RF patent for utility model No. 193927), *EV*	Difference, *times*
10	6.1×10^4	1.1×10^4	5.5
100	5.4×10^7	6.5×10^6	8.3
1000	5.4×10^{10}	6.1×10^9	8.8

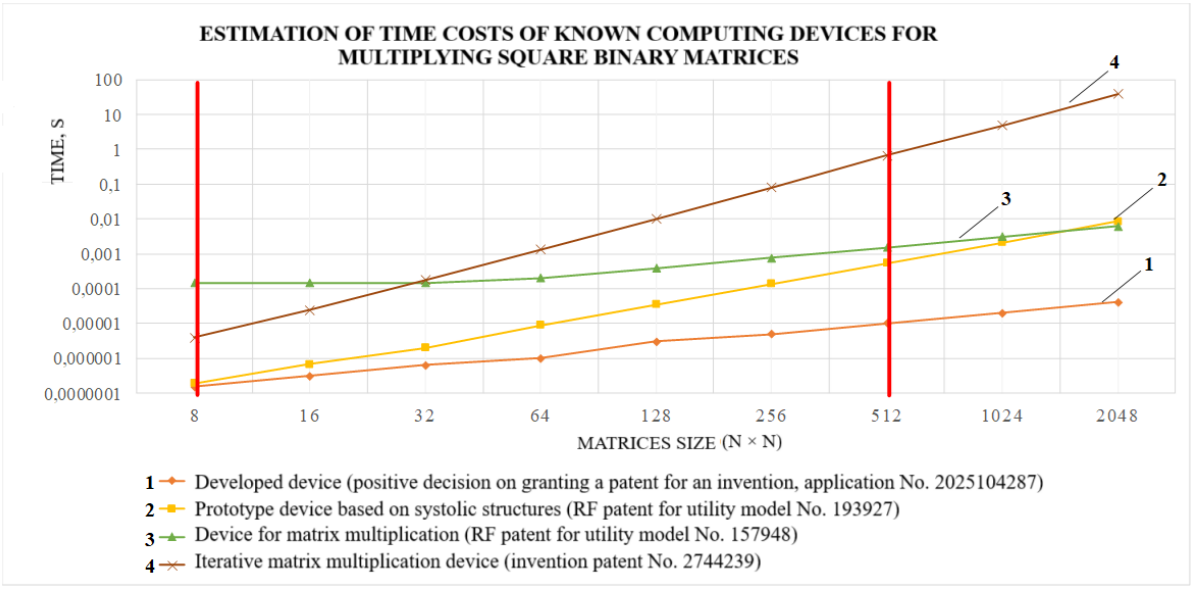

Fig. 9 Estimation of the Time Spent by Known Computing Devices on the Multiplication of Binary Matrices.

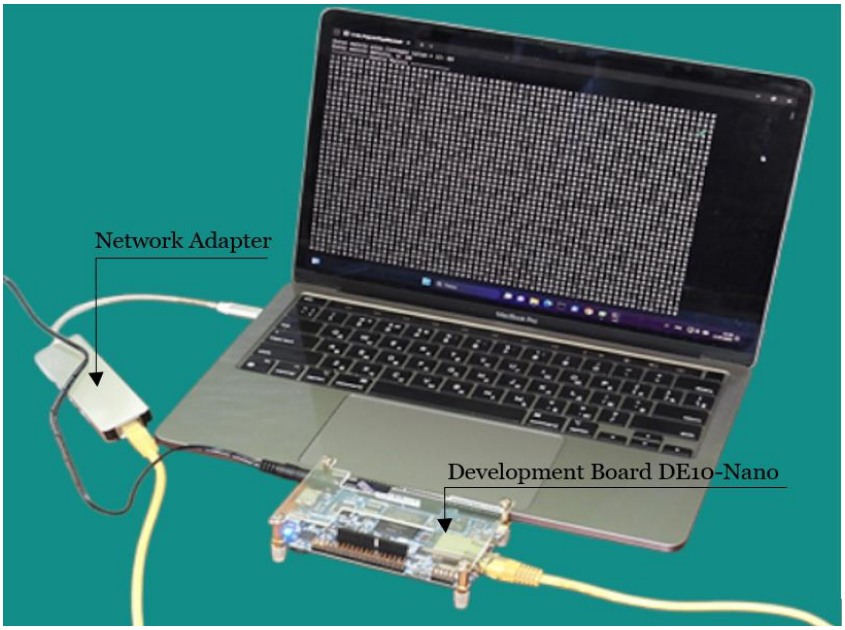

Fig. 10 Practical Implementation of the Developed Specialised Computing Device Using an FPGA.

Fig. 11 Practical Implementation of the Developed Specialised Computing Device Using an FPGA.

The DE10-Nano board offers numerous features that enable users to implement a range of designed circuits. Figure 12 shows the block diagram of the DE10-Nano board. All connections are established via the Cyclone V SoC FPGA, providing maximum flexibility for users (see the DE10-Nano user manual). In addition, by slightly modifying the storage unit cell structure, the developed computing device can perform general-purpose matrix operations.

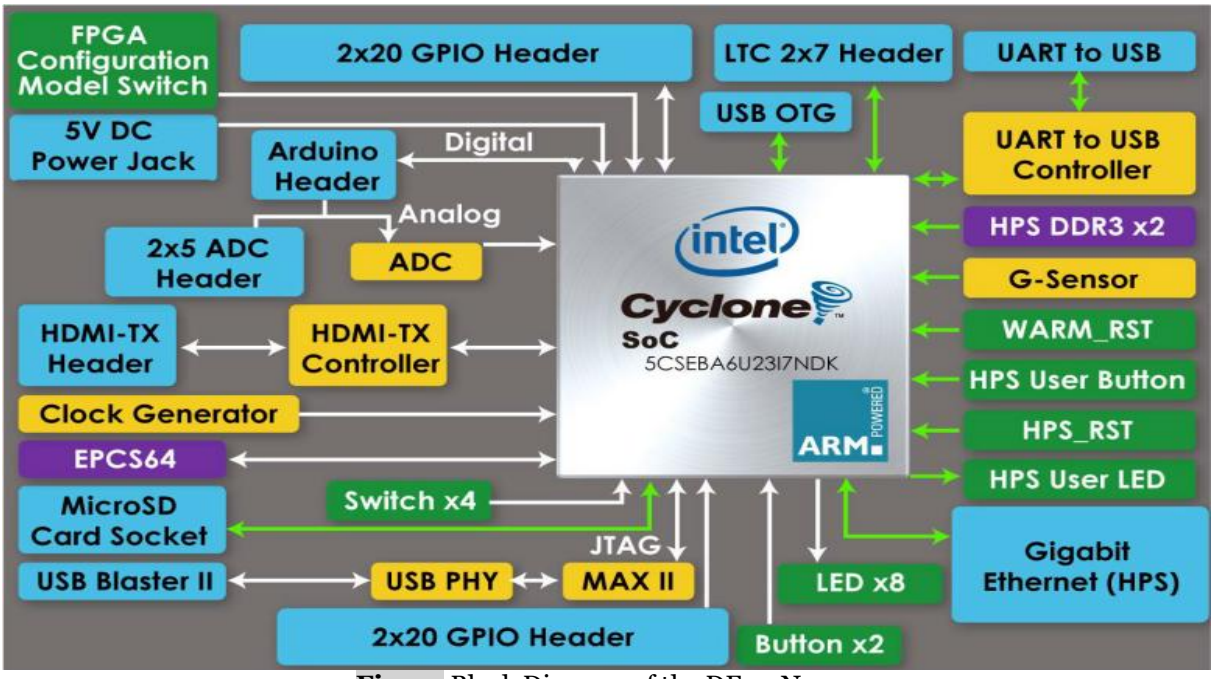

Fig. 12 Block Diagram of the DE10-Nano.

4.CONCLUSIONS

Even though the developed specialized computing device for multiplying square binary matrices has a slightly higher hardware complexity than the prototype device, according to preliminary estimates, it reduces time costs by up to 52.4 times when multiplying square binary matrices with a size of $n \leq 512$, which is advisable for its practical implementation in a semi-custom design using FPGA or in a customized version using ASIC. Our results indicate promising prospects for future innovation and research. It can play a significant role across broad scientific fields by exploring the integration of our device with advanced processor technologies, such as quantum computing and neuromorphic systems, thereby providing an exciting avenue for future work and innovation in matrix computation. It could also play a key role in AI algorithms that use sensor data, such as the Pelican Optimisation Algorithm (POA) and Particle Swarm Optimisation (PSO) [33-35]. This work presents a novel systolic device whose core innovation is a pipelined multiport memory architecture. This architecture directly addresses and overcomes the critical data bandwidth bottleneck, the primary obstacle to efficiency in conventional designs. By ensuring a continuous, high-bandwidth data flow to the processing elements, our contribution enables the systolic array to operate at its theoretical peak performance. Thus, the proposed device represents a significant advancement, offering a scalable and efficient solution for the most computationally intensive tasks.

REFERENCES

[1] Zykov AG, Polyakov VI. **Algorithms for Computer-Aided Design of Electronic Computing Machines.** *Saint Petersburg: ITMO University* 2014; 136.

[2] Saenko IB, Mityakov ES, Lauta OS, Sokolov AP. **Swarm Control Algorithm for UAVs with Elements of Cluster Analysis.** *Information and Space* 2024; **4**: 68–75.

[3] Volkov AS, Baskakov AE. **Development of a Bidirectional Search Procedure for Solving the Routing Problem in Software-Defined Transport Networks.** *Trudy MAI (Proceedings of Moscow Aviation Institute)* 2021; **118**: 1-15.

[4] Zykov AA. **Fundamentals of Graph Theory.** *Moscow: Nauka Publishing* 1986; 384.

[5] Vatutin EI, Zotov IV. **Construction of a Relation Matrix in the Problem of Optimal Partitioning of Parallel Control Algorithms.** *Izvestiya of Kursk*

State Technical University 2004; **2**: 85–89.

[6] Levitin AV. **Overcoming Limitations: The Bisection Method.** *Algorithms: Introduction to Design and Analysis* 2006; 349–353.

[7] Ismail H, Toroslu IH. **Improving the Floyd–Warshall All-Pairs Shortest Paths Algorithm.** *Department of Computer Engineering, METU* 2021; 1–5.

[8] Bolgak AV, Vatutin EI. **Evaluation of the Actual Performance of Intel Core Processors of Various Generations in the Task of Real Matrix Multiplication for Single-Threaded Software Implementation.** *Cloud and Distributed Computing Systems in E-Government* 2024; 98–100.

[9] Vatutin EI, Martynov IA, Titov VS. **Evaluation of the Real Performance of Modern GPUs Supporting CUDA Technology in the Matrix Multiplication Problem.** *Izvestiya of Southwest State University. Series: Control, Computer Engineering, Informatics, Medical Instrumentation* 2014; **2**: 8–17.

[10] Boreskov AV, Kharlamov AA, Markovsky ND, Mikushin DN, Mortikov EV, Myl'tsev AA, Sakharnykh NA, Frolov VA, Sadovnichiy VA. **Parallel Computing on GPUs: Architecture and CUDA Programming Model.** *Moscow University Press* 2012; 336.

[11] Starovoitov IN, Revnyakov EN, Polyakova EN. **Parallel Computing on Graphics Processors.** *Proceedings of the First International Scientific Conference on Digitalisation Issues: EDCRUNCH URAL 2020* 2020; 314–319.

[12] Khaldoon AO, Kamil NY, Abbas A, Al Smadi T. **A Novel Flying Robot Swarm Formation Technique Based on Adaptive Wireless Communication Using a MUSIC Algorithm.** *International Journal of Electrical and Electronics Research* 2024; **12**(2): 688–695.

[13] Yushin AM. **Optoelectronic Devices and Their Foreign Analogues: Reference Book.** *Moscow: RadioSoft Publishing* 2000; **1**: 512.

[14] Trrad I, Smadi TA, Al-Wahshat H. **Application of Fuzzy Logic to Cognitive Wireless Communications.** *International Journal of Recent Technology and Engineering* 2019; **8**(3): 2228–2234.

[15] Plaksienko VS, Plaksienko NE, Plaksienko SV, Plaksienko VS. **Signal Reception and Processing Devices: A Textbook for Universities.** *Moscow: Educational and Methodological Publishing Centre "Uchebnaya Literatura"* 2004; 376.

[16] Al-Maitah M, Al Smadi TA, Al-Zoubi HQR. **Scalable User Interface.** *Research Journal of Applied Sciences, Engineering and Technology* 2014; **7**(16): 3273–3279.

[17] Gümüşkaya H, Örencik B. **A Parallel Pipelined Computer Architecture for Digital Signal Processing.** *Turkish Journal of Electrical Engineering and Computer Sciences* 1998; **6**(2): 107–130.

[18] Strogonov AV. **Fundamentals of Digital Signal Processing.** *Voronezh: Voronezh State Technical University* 2014.

[19] Gvozdeva SN, Vatutin EI, Pshenichnykh AO, Titov VS. **Device for Multiplying Binary Matrices.** *Patent of the Russian Federation for Utility Model No. 193927* 2019.

[20] Gvozdeva SN, Vatutin EI, Titov VS. **Performance Evaluation of a Systolic Structure Device for Binary Matrix Multiplication.** *Telecommunications* 2020; **3**: 2–10.

[21] Bolgak AV, Vatutin EI. **Device for Multiplying Binary Matrices.** *Patent for Invention No. 2025104287* 2025.

[22] Bolgak AV, Vatutin EI. **Mathematical Model and Structural–Functional Organization of a Parallel–Pipelined Memory Device for Fast Multiplication of Square Binary Matrices.** *XXI Century: Results of the Past and Problems of the Present Plus* 2025; **14**(2): 27–34.

[23] Bolgak AV, Vatutin EI, Trokoz DA. **Estimation of Time Costs for Multiplying Square Binary Matrices in a Device with Pipelined Data Reading from Specialised Multi-Port Memory.** *Izvestiya Southern Federal University. Engineering Sciences* 2025; **4**(246): 6–20.

[24] Al-Maitah M, Al Smadi TA, Al-Zoubi HQR. **Scalable User Interface**. *Research*

Journal of Applied Sciences, Engineering and Technology 2014; 7(16): 3273-3279.

[25] Smadi TAA. **Computer Application Using a Low-Cost Smart Sensor.** *International Journal of Computer Aided Engineering and Technology* 2012; **4**(6): 567.

[26] Vatutin EI, Titov VS. **Evaluation of the Actual Performance of Modern Processors in the Matrix Multiplication Problem for Single-Threaded Software Implementation Using SSE Extension.** *Izvestiya of Southwest State University* 2015; **1**(4): 26–35.

[27] Vatutin EI, Martynov IA, Titov VS. **Evaluation of the Actual Performance of Modern GPUs Supporting CUDA Technology in the Matrix Multiplication Problem**. Izvestiya of Southwest State University. Series: Control, Computer Engineering, Informatics, Medical Instrumentation. 2014;2:8–17. (in Russian).

[28] Martynov IA, Vatutin EI, Titov VS. **Device for Matrix Multiplication.** *Patent of the Russian Federation for Utility Model No. 157948* 2015.

[29] Gvozdeva SN, Vatutin EI, Titov VS. **Device for Use During a Binary Matrix.** *Patent of the Russian Federation No. 2744239* 2021.

[30] Bolgak AV, Vatutin EI. **Assessment of the Hardware Complexity of a Device for Multiplying Square Binary Matrices with Pipelined Data Reading from Specialised Multi-Port Memory.** *Trudy MAI (Proceedings of Moscow Aviation Institute)* 2025; **143**: 1-20.

[31] Gribok S, Pasca B. **Efficient 8-Bit Matrix Multiplication on Intel Agilex-5 Fogas.** *IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines* 2024; 43–53.

[32] Nayak SK, Nayak AK, Laha SR, Tripathy N, Smadi TA. **A Robust Deep Learning-Based Speaker Identification System Using a Hybrid Model on the KUI Dataset.** *International Journal of Electrical and Electronics Research* 2024; **12**(4): 1502–1507.

[33] Khaleel RZ, et al. **Improved Trajectory Planning for a Mobile Robot Based on the Pelican Optimization Algorithm.** *Journal Européen des Systèmes Automatisés* 2024; **57**(4): 1125-1134.

[34] Khaleel HZ, Humaidi AJ. **Toward Accuracy Improvement in the Solution of the Inverse Kinematic Problem in a Redundant Robot: A Comparative Analysis.** *International Review of Applied Sciences and Engineering* 2024; **15**(2): 242–251.

[35] Obied H, et al. **Implementation and Derivation Kinematics Modeling Analysis of WidowX 250 6 Degrees of Freedom Robotic Arm.** *Journal of Engineering and Applied Sciences* 2024; **19**: 45-56.