FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 32, No 4, December 2019, pp. 479 - 501
https://doi.org/10.2298/FUEE1904479D

PARALLEL MATRIX MULTIPLICATION CIRCUITS FOR USE IN KALMAN FILTERING

Rafal Długosz, Katarzyna Kubiak,
Tomasz Talaśka, Inga Zbierska-Piątek

[1]UTP University of Science and Technology
Faculty of Telecommunication, Computer Science and Electrical Engineering
[2]Adam Mickiewicz University
Faculty of Mathematics and Computer Science
[3]Aptiv Services Poland

Abstract. *In this work we propose several ways of the CMOS implementation of a circuit for the multiplication of matrices. We mainly focus on parallel and asynchronous solutions, however serial and mixed approaches are also discussed for the comparison. Practical applications are the motivation behind our investigations. They include fast Kalman filtering commonly used in automotive active safety functions, for example. In such filters, numerous time-consuming operations on matrices are performed. An additional problem is the growing amount of data to be processed. It results from the growing number of sensors in the vehicle as fully autonomous driving is developed. Software solutions may prove themselves to be insuffucient in the nearest future. That is why hardware coprocessors are in the area of our interests as they could take over some of the most time-consuming operations. The paper presents possible solutions, tailored to specific problems (sizes of multiplied matrices, number of bits in signals, etc.). The estimates of the performance made on the basis of selected simulation and measurement results show that multiplication of 3×3 matrices with data rate of 20 100 MSps is achievable in the CMOS 130 nm technology.*

Key words: *Matrix multiplication, Parallel circuits, asynchronous solutions, Kalman filter, automotive applications, CMOS.*

Received July 1, 2019
Corresponding author: Rafal Długosz
UTP University of Science and Technology, Faculty of Telecommunication, Computer Science and Electrical Engineering, Bydgoszcz, Poland
(E-mail: rafal.dlugosz@gmail.com)

1 Introduction

An increasing number of systems and applications requires advanced mathematical computation as well as processing of large amounts of data. Such a situation happens even in the areas which traditionally formerly were not associated with such topics as artificial intelligence (AI), signal processing or data mining. For example, in the automotive industry, widely understood signal processing became one of the key development areas in algorithms used in Advanced Driver Assistance Systems (ADAS) in Active Safety (AS) area [1–3].

An observed rapid development of the AS systems towards more complex solutions related to autonomous driving causes several problems. One of them is the computing power required to process increasing amount of data from the vehicle's environment. A second problem is how to minimize the energy consumed by the increasing number of the applied computing devices. Yet another problem is also the so-called business case, i.e. the necessity to reduce costs of the overall system. This leads to trade-offs between the computation power, that has to be sufficient to process increasing amount of data collected by vehicle on-board sensors [4] and the price of the devices, in which the computing algorithms are implemented.

An additional challenge, common in ADAS functions, is the real time data processing [5]. New data scans, for example from radar and camera sensors, are provided to the AS system every 30-50 ms. The overall signal processing, as well as the decision (control strategy) made on the basis of the output signals from the ADAS functions have to be completed in this short period of time due to safety reasons and general system performance, for example. One of the issues in real-time operating systems (RTOSs) in vehicles equipped with the AS functions, is how to effectively manage the computational power [6]. Depending on the complexity of the performed operations, the overall signal processing scheme may be divided into blocks. Some tasks can then be implemented on a set of supporting devices. One can also consider developing specialized integrated circuits (ASIC – application specific integrated circuit), optimized for selected tasks. This is the topic of this work.

Since 1960 when R.E. Kalman published his well-known paper [7], his filter (named Kalman) has gained large popularity. This recursive solution to discrete data linear filtering problem has become the subject of numerous investigations due to its wide application in digital computing. Because of its properties, the algorithm is widely used in different areas that include, for example, speech enhancement [8], noise reduction in autopilot in autonomous boats [9] or in enhancement of robot path [10]. It is worth to mention that Kalman filter is frequently applied in the localization and map building problem (SLAM) [11].

In the automotive AS applications one of the basic operations is tracking objects present in the vicinity of the moving vehicle [12]. The role of the tracker is to build an image of a real object on the basis of collected radar and vision data. The image means such features as the type of the object (pedestrian, bicycle, another vehicle), its size, position, spatial trajectory, velocity, etc. The Kalman filters are commonly used in such algorithms [13]. The operation which consumes the most computational power in such filters is the multiplication of matrices. For this reason, this work focuses on hardware realizations of the circuits that may speed up such operations. The circuits of this type have to be flexible. This means that they should be scalable and thus suitable for different sizes of the multiplied matrices. In our opinion, fully parallel and fully asynchronous data processing is difficult to achieve. However, a combination of parallel and serial methods leads to the best results in terms of data rate and the circuit complexity.

The paper is organized as follows. In the next section state-of-the-art study in the related topics is presented. In the following section the proposed solutions are introduced and briefly discussed in the context of specific values of other parameters. Then, verification of selected circuits (simulations and measurements) is demonstrated. Conclusions are formulated in the last section.

2 State-of-the-art

In this section two aspects of the presented problem are described. One of them is a brief review of selected solutions for Kalman filtering and the complexity of matrix multiplication operations required in a given case. Such a study is necessary to estimate the requirements for the structure of the ASIC coprocessor for matrix multiplication, if this solution is to be a flexible one.

2.1 Matrix multiplication

Matrix multiplication is one of the important stages in many applications. In numerical algebra a lot of problems come down to the matrix multiplication [14]. Even in complex structures such as multi-core systems the matrix multiplication plays an important role [15]. Exemplary investigations in which the matrix multiplication is applied are presented in [16]. Authors of this publication propose a distributed computing accelerator based on parallel random access memory (RRAM) crossbar for the matrix multiplication. The main idea is to use the matrix vector multiplication to multiple inner operations of two vectors. Each of these operations is produced by one RRAM crossbar. These studies revealed that the binary matrix multiplication can be done with smaller memory sizes, lower computing delay and higher energy efficiency using this approach.

The operations of the matrix multiplication are in common use in Kalman filtering. This type of filter has numerous applications which include speech enhancement [8], noise reduction in autopilot in autonomous boats [9], and creating and tracking of objects on the basis of radar scans [17]. The last feature is more and more frequently used in the automotive industry in Advanced Driver Assistance Systems (ADAS). Kalman filter plays an important role in creation of objects on the basis of radar data – reflections provided as a set of pairs of the azimuth and the range values. The objects (tracks) become an input to numerous active safety functions. They include, for example, adaptive cruise control (ACC), in which crucial aim is to obtain nearly perfect tracking algorithm performance. Otherwise, the system could mislead the end user or even not be able to protect from an accident.

The presented examples of the application of the matrix multiplication lead to a question of an efficient implementation of such operations. One of the important aspects here is the flexibility which means the possibility of reconfiguration of the circuit. Other important factors include the computation speed, the power dissipation and the chip area. The last feature is relevant in the case of CMOS realizations in the form of application specific integrated circuits (ASIC) [18]. Such implementations as well counterpart realizations in Field Programmable Gate Arrays (FPGA) [19, 20] allow for massive parallel data processing. This is crucial in overcoming the shortage of the computational power in advanced active safety functions.

FPGA platforms are frequently used in such applications that require parallel data processing and where a reconfiguration of the in virtually every moment. In the case of ASIC specialized systems, it is not possible to change the layout topology after their physical implementation. However, the undoubted advantage is the possibility of obtaining better technical parameters, such as reduction of energy consumption, increase of work speed, construction of any atypical asynchronous blocks, etc. All these advantages cause that these systems are also often used for the implementation of various complex, demanding fast and energy-saving algorithm work [18].

In the literature several solutions in which matrix multiplication is realized as ASIC are described. One of them can be found in [21]. This is a processor working in real time with a dynamically reconfigured 4×4 output matrix. Particular elements of this matrix are calculated in parallel by means of 16 multiplication circuits. However, particular elements of the output matrix are calculated sequentially. In subsequent clock cycles a single multiplication and accumulation operation is performed for each of the multipliers.

A similar approach as in [21] can be found in [22]. The circuit was tested in the CMOS 90 nm technology. Following elements of the input matrices are provided to the inputs through a multiplexer. Then, they are multiplied by themselves and accumulated. In this approach, an improvement in the achieved data rates was possible by rearranging the matrix element into a 2-dimensional array of processed elements interconnected as a mesh. The edges of each row and column are interconnected in a torus structure.

In [23] a partially analog solution with a vector-by-matrix computation technique implemented in the 55 nm technology is proposed. In this case, multi-bit input and output signals are represented by time-encoded digital signals, while multi-bit matrix weights are realized with current sources. One of the important advantages of this solution is compact peripheral circuits which allow for energy savings. As a case study, the authors of [23] realized a multilayer perceptron, based on two layers of 10×10 four-quadrant multipliers.

Another circuit with partially analog signal processing has been reported in [24]. It is a charge-mode parallel structure which allows for vector by matrix multiplication. The presented solution has internally analog and externally digital architecture. A unit cell used here combines single-bit dynamic random-access memory and a charge injection device binary multiplier as well as an analog accumulator.

A more detailed comparison of described ASIC solutions is presented in Table 2.1.

2.2 Kalman filtering in automotive applications in brief

The Kalman filter is an example of a tool which predicts the future state of a system basing on the estimate of its current state. It is modeled by the following state equations:

Table 1: Comparison of selected matrix multiplication circuits realized in the CMOS technology, at the transistor level

Ref	CMOS	sizes	NBI	Speed	Power or energy
[21]	0.25 μm	4×4	16	18.8ns	NA
[22]	90 nm	4×4	4	2 μs	3.12 mW
[23]	55 nm	10×10	6	$150\cdot10^9$ OP/J	7 fJ / OP*
[24]	0.5 μm	512×128	8	2×10^{12} MAPS	0.5 pJ / MA

MAPS	–	multiply-accumulation operations per second
MA	–	multiply-accumulation operation (OP)
(*)	–	for the sizes of matrices > 200

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{u}(n) + \mathbf{v}(n), \tag{1}$$

$$\mathbf{y}(n+1) = \mathbf{C}\mathbf{x}(n) + \mathbf{w}(n), \tag{2}$$

where: n are discrete time moments (following samples)
A is the state transition matrix, applied to a previous state ($\mathbf{x}(n)$)
B is the control input matrix
C is the output matrix
v is the processing noise vector
w is the measurement noise vector
y is the measurement vector

The calculation scheme in the Kalman filter requires multiplication of matrices and vectors. Equations (1) and (2) do not contain all required operations. The additional operations may include the calculation of various covariance matrices [25] and their number depends on the problem to be solved.

3 Proposed matrix multiplication solutions

At the transistor level, the operation of the matrix multiplication may be implemented in various ways. They include parallel and serial solutions, operating in parallel or in serial fashion. Data processing may be performed asynchronously or sequentially. In practice, the optimization of the structure of the matrix multiplication circuit will largely depend on the sizes of the input signals (matrices), the resolution of the multiplied numbers (the

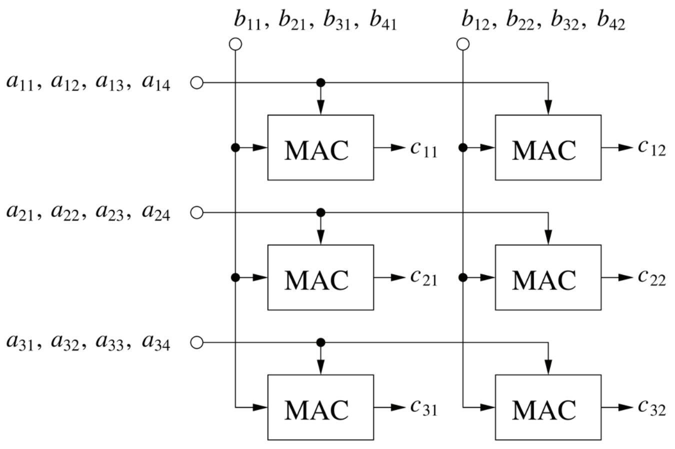
$b_{11}, b_{21}, b_{31}, b_{41}$
$b_{12}, b_{22}, b_{32}, b_{42}$
$a_{11}, a_{12}, a_{13}, a_{14}$
$a_{21}, a_{22}, a_{23}, a_{24}$
$a_{31}, a_{32}, a_{33}, a_{34}$
MAC
c_{11}
c_{12}
c_{21}
c_{22}
c_{31}
c_{32}

Fig. 1: Diagram of an exemplary parallel matrix multiplication circuit.

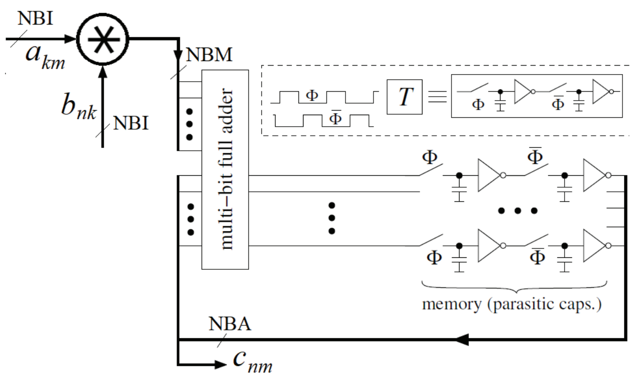

Fig. 2: The MAC circuit applied for the matrix multiplication.

number of bits), the required data rate and the importance of low energy consumption. Such factors as the number of external pads available may also impact the implementation approach. Depending on these parameters, various mixed approaches may be proposed.

In principle, the sequential (serial) approach corresponds to the implementations found in software systems. In general, it uses a single multiplier and an accumulator (ACU) and calculates particular elements of the output matrix in several nested loops. Similarly, the matrix multiplication can be also realized at the transistor level using a multiphase clocks and an appropriate commutation field composed of switches or logic gates. In this case, the speed of the circuit, understood as the rate of generating the whole resultant matrix at its output, depends inversely on the sizes of the input matrices. Therefore, a fully serial approach is not optimal in terms of performance, however, it is essentially optimal in terms of the circuit complexity, i.e. the number of used transistors. In this approach, the circuit complexity depends on the sizes of the input matrices to a relatively small extent.

Parallel matrix multiplication, which is the subject of this work, can be implemented in various ways. Generally, multiplication of two-dimensional matrices $A_{N_\mathrm{A},M_\mathrm{A}}$ and $B_{N_\mathrm{B},M_\mathrm{B}}$ returns a two-dimensional matrix $C_{N_\mathrm{C},M_\mathrm{C}}$ at the circuit output. Indices N_A, M_A, N_B, M_B, N_C, M_C are the sizes of particular matrices, where N and M denote rows and columns, respectively. The following relations occur between these parameters: $M_\mathrm{A} = N_\mathrm{B} = K$, $N_\mathrm{C} = N_\mathrm{A}$, $M_\mathrm{C} = M_\mathrm{B}$

Each of the elements of the C matrix (c_{nm}) is the sum of a given number (K) of the products of respective components of the input matrices A and B. In the iterative approach, all elements of the output matrix C are calculated on the basis of the accumulation of products in subsequent steps of the loop. In this situation, the parallelism can be understood as simultaneous calculation of all elements of the output matrix. However, particular multiplication and accumulation operations for each output element c_{nm} are sequentially performed in a given number of iterations, K, where $K = M_\mathrm{A} = N_\mathrm{B}$. Such a solution is illustrated in Fig. 1, for the multiplication operation of exemplary matrices $A_{3,4}$ and $B_{4,2} \to C_{3,2}$:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix} \tag{3}$$

The main component in the described parallel-iterative approach is the multiplication-accumulation channel (MAC). A general structure of this circuit is shown in Fig. 2. The circuit is composed of a multiplication circuit and an ACU block. Possible realizations of the MAC circuit are presented in more detail below. In an exemplary case given by 3, the output matrix C consists of 6 elements c_{nm}, where n and m denote the position of a given element (row, column) in matrix C. In this case, six MACs are used, while the computation of particular elements of the output C matrix requires 4 multiplication-accumulation iterations.

It is worth to notice that the parameters M_A and N_B can theoretically have any value ≥ 1 for a given number of MACs and their arrangement at the circuit output. For the exemplary matrix $C_{3,2}$ as above, the following operation may be performed:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1M_\mathrm{A}} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2M_\mathrm{A}} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3M_\mathrm{A}} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ \dots & \dots \\ b_{N_\mathrm{B}1} & b_{N_\mathrm{B}2} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix} \tag{4}$$

In practice, the values of the M_A and N_B parameters are limited by the resolution of the ACU and the structure of the clock generator, i.e. the number of clock phases. The number of bits in the ACU should be sufficiently large so that the product of two largest numbers of the same sign can be accumulated and the ACU overflow can be avoided.

3.1 Components of the MAC block

In generak, the MAC circuit is composed of three subcircuits. Two of them – a multi-bit full adder (MBFA) and the memory block – are parts of the ACU block. The third one, the multiplier, may be a separate circuit or it may also be integrated with the ACU in some situationcs. Selected solutions for these components are presented below.

3.1.1 Multi-bit full adder

The MBFA is an asynchronous circuit, composed of a chain of 1-bit full adders (1BFA) [26] coupled through the carry in and carry out terminals. Various 1BFAs circuits have been proposed in the literature [27–29]. For this reason, the 1BFA and the MBFA circuits are treated as standard blocks and are not described in more detail in this work.

The required size of the MBFA depends on the signal resolution at the output of the multiplier (NBM) as well as the assumed number of iterations K, and can be expressed as follows:

$$\mathrm{NBA} = \mathrm{NBM} + \log_2 K + d, \tag{5}$$

where d equals 0 for $K \in \{2, 4, 8, 16, ...\}$ and 1, otherwise. The value of NBM equals 2·NBI, where NBI is the number of bits in the input a and b signals.

3.1.2 Memory block

The memory block may be realized in several ways. Fig. 2 presents one of the possible solutions, in which two switches and two NOT gates are used per a single bit of the signal stored in the ACU. In this case, the values of particular bits are stored in parasitic gate-to-source capacitances C_{GS} of the NOT gates. The overall memory block is controlled by a two-phases clock, which updates the memory after the summation operation and provides the resultant value to one of the inputs of the MBFA. In an alternative approach, a block of D-flip flops (DFF) instead of NOT gates and switches may me used.

Both these solutions offer some advantages. The first approach requires a smaller number of transistors and consumes much less energy. Since the switches are realized as transmission gates composed of the PMOS and NMOS devices, a single bit in the memory requires only 8 transistors. By comparison, a typical DFF is built of 26 transistors (6 NAND gates in a chain of RS latches). On the other hand, the C_{GS} capacitances, used in the first approach, are short-range memory cells which may lose the stored information if the clock frequency is too small due to the leakage effect. By comparison, the DFFs offer theoretically infinite storage time. For this reason, the second approach is more suitable for low data rates.

In the prototype chip designed by us, in which the MAC block is one of the components, the first approach was used consciously. The aim was to check this concept as we also implemented a 10-phases control clock [30] which works on a similar basis. In the clock circuit, the chain of the NOT gates and the switches was much longer than in the described memory (20 NOT gates and 20 switches). This approach allowed to limit the energy consumption to a level of only about 5 % of the energy that would have been consumed in the case of using the DFFs. In short lines like in the memory described above, the second option seems to be a better solution, because it requires only a 1-phase clock.

3.1.3 Multiplication operation

The way in which the multiplication circuit is implemented impacts the complexity of the structure of the MAC block and the controlling clock generator, as well as the speed of the overall circuit. Let us consider two possible approaches. The first one corresponds to a conventional shift-and-add operation which is similar to the ACU circuit described above to some extent. It is composed of a single MBFA, a shift register, a memory block and a set of AND gates. One of the input signals is shifted along the register in subsequent clock cycles. It can be thought of as increasing the marker of the position of this signal in the register by 1 in each clock phase. Then, the resultant signal is added (or not) to the ACU, depending on the values of corresponding bits of the second input signal. These bits mask the signal in the register throughout the AND gates.

In the second approach, a fully asynchronous multiplier is realized as a binary tree composed of a larger number of MBFAs of different sizes. This solution is also based on the shift-and-add principle. However, particular shifting and summing operations are performed in a parallel and asynchronous manner without the use of the controlling clock generator in this case.

Both these approaches offer some unique advantages and suffer from some limitations. The first one requires smaller number of transistors, as only one MBFA is needed. However, the maximum data rate is inversely proportional to the signal resolution. The asynchronous approach, on the other hand, is much faster as the duration of the overall multiplication operation results only from delays introduced by asynchronous MBFAs at particular layers of the tree. The number of layers (N_{BTL}) increases only moderately with the number of bits of particular input signals (NBI), which is an assest. Assuming that NBI belongs to a set of powers of the number 2 (NBI $\in \{2, 4, 8, 16, ...\}$), the N_{BTL} factor equals:

$$N_{\mathrm{BTL}} = \log_2 \mathrm{NBI}. \tag{6}$$

For exemplary signal resolutions of 8, 16, 32 bits, the number of layers equals 3, 4 and 5, respectively. For the circuit implemented in the CMOS 130 nm technology, the multiplication time does not exceed 5, 7 and 9 ns, respectively, i.e. it increases only moderately. Moreover, the asynchronous approach does not need an additional clock circuit to control the multiplier. Higher circuit complexity is the price for these advantages.

One of the advantages of the first approach is the possibility of using a single MBFA and a single memory block, which additionally simplifies the structure of the MAC. The overall circuit may be controlled by a single clock generator but with a substantially more complex structure. In this case, the number of clock phases is NBI times larger than for the asynchronous approach:

$$N_{\mathrm{Sclk}} = \mathrm{NBI} \cdot k. \tag{7}$$

This solution requires an appropriate commutation field which supplies particular elements of the input matrices A and B to appropriate inputs of the ACU block. One of the technical problems is the fact that the marker of the position has to be shifted to the beginning of the shift register after every subsequent k clock phase. This is due to the fact that new elements of the input matrices are provided at these time instants to the inputs of the ACU, with one of them uploaded in the register.

At the level of a single MAC block, various mixed solutions can be introduced. To show one of them, the calculation of a single element of the C matrix is considered. To simplify the explanation, let us consider the multiplication operation of vectors A and B, as below. In this example $N = M$ is formally assumed:

$$\begin{bmatrix} c \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_M \end{bmatrix} \times \begin{bmatrix} b_1 & b_2 & b_3 & \dots & b_N \end{bmatrix}^{\mathrm{T}}. \tag{8}$$

In a more general form is may be expressed as:

$$c_{nm} = \sum_{k=1}^{K} a_{km} \cdot b_{nk}. \tag{9}$$

It is assumed that the numbers shifted in the register are elements of the A vector, while the masking signals are particular bits of the corresponding elements of the B vector. A single element, b_n, of the B vector can be expressed as follows:

$$b_n = 2^0 \cdot b_{n,1} + 2^1 \cdot b_{n,2} + 2^2 \cdot b_{n,3} + \dots + 2^{L-1} \cdot b_{n,L}, \tag{10}$$

where $b_{n,l}$ are following bits of this element. The $b_{n,1}$ and $b_{n,L}$ values are the least significant and the most significant bits (LSB and MSB), respectively. On the basis of 10 the operation 8 can be expressed as follows:

$$\begin{aligned} [c] \quad = \quad & 2^0 \cdot a_1 \cdot b_{1,1} + 2^1 \cdot a_1 \cdot b_{1,2} + \dots + 2^{L-1} \cdot a_1 \cdot b_{1,L} \\ & 2^0 \cdot a_2 \cdot b_{2,1} + 2^1 \cdot a_2 \cdot b_{2,2} + \dots + 2^{L-1} \cdot a_2 \cdot b_{2,L} \\ & \dots \\ & 2^0 \cdot a_M \cdot b_{N,1} + 2^1 \cdot a_M \cdot b_{N,2} + \dots + 2^{L-1} \cdot a_M \cdot b_{N,L} \end{aligned} \tag{11}$$

Equation 11 can be reorganized as follows:

$$\begin{aligned} [c] \quad = \quad & 2^0 \cdot (a_1 \cdot b_{1,1} + a_2 \cdot b_{2,1} + \dots + a_M \cdot b_{N,1}) + \\ & 2^1 \cdot (a_1 \cdot b_{1,2} + a_2 \cdot b_{2,2} + \dots + a_M \cdot b_{N,2}) + \\ & 2^2 \cdot (a_1 \cdot b_{1,3} + a_2 \cdot b_{2,3} + \dots + a_M \cdot b_{N,3}) + \\ & \dots + \\ & 2^{L-1} \cdot (a_1 \cdot b_{1,L} + a_2 \cdot b_{2,L} + \dots + a_M \cdot b_{N,L}) \end{aligned} \tag{12}$$

After replacing the multiplication operation (2^l factors) with the bit-shifting operations, as in the shift-and-add circuit, formula 12 can be expressed as follows:

$$\begin{aligned}[c] \;=\; & (a_1 \cdot b_{1,1} + a_2 \cdot b_{2,1} + \cdots + a_M \cdot b_{N,1}) \ll 0+ \\ & (a_1 \cdot b_{1,2} + a_2 \cdot b_{2,2} + \cdots + a_M \cdot b_{N,2}) \ll 1+ \\ & (a_1 \cdot b_{1,3} + a_2 \cdot b_{2,3} + \cdots + a_M \cdot b_{N,3}) \ll 2+ \\ & \cdots + \\ & (a_1 \cdot b_{1,L} + a_2 \cdot b_{2,L} + \cdots + a_M \cdot b_{N,L}) \ll L \end{aligned} \tag{13}$$

In this approach, the summation operations in the particular lines of 13 above may be performed asynchronously, e.g. in a binary tree. Only the results of this summation are uploaded to the register. The number of clock phases can thus be reduced to NBI. The circuit is also around K times faster than in the first (fully sequential) approach.

3.2 Summary

To facilitate the discussion let us denote particular solutions as follows:

Solution 1 (S1): Sequential (serial) approach with a single MAC: (S1.a) serial multiplication circuit, (S1.b) parallel and asynchronous multiplication circuit based on a binary tree.

Solution 2 (S2): Parallel approach with the number of the MACs equals the number of elements in the matrix C: (S2.a) serial multiplication circuit, (S2.b) parallel and asynchronous multiplication circuit based on a binary tree.

Solution 3 (S3): Various mixed approaches.

4 Verification of the proposed solutions

Selected results showing the performance of the proposed circuits are presented in this section. It has been shown that the parallel matrix multiplication can be decomposed into computational channels, in which the values of particular elements of the output matrix C are determined independently. For this reason, here we focus on the operation of a single MAC block. The performance assessed in this way can then be used to determine the parameters of the overall circuit.

In the previous section, two approaches to the implementation of the memory and the ACU blocks were presented. In the first on a block of

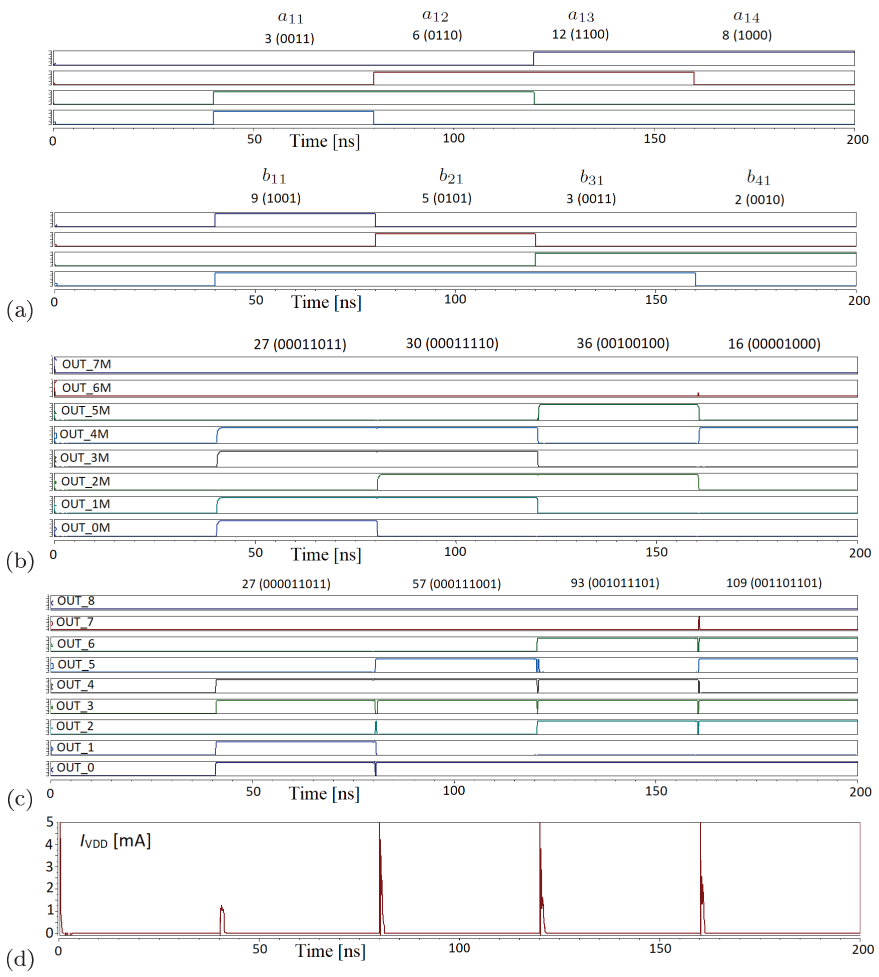

Fig. 3: Selected simulation results: (a) selected input signals a_{1m} and b_{n1} from Table 4.1, (b) binary outputs of the asynchronous multiplier, (c) binary outputs of the MBFA that is also the output signal of the overall MAC, and (d) the supply current.

the chains of the NOT logic gates and the switches is used, while in the second one the memory is realized using D flip-flops. The first solution was implemented in a prototype integrated circuit fabricated in 130 nm CMOS technology and verified by both simulations and measurements. The second approach was verified by simulations in the HSpice environment in TSMC CMOS 180 nm technology.

An asynchronous and parallel multiplier were used in both cases.

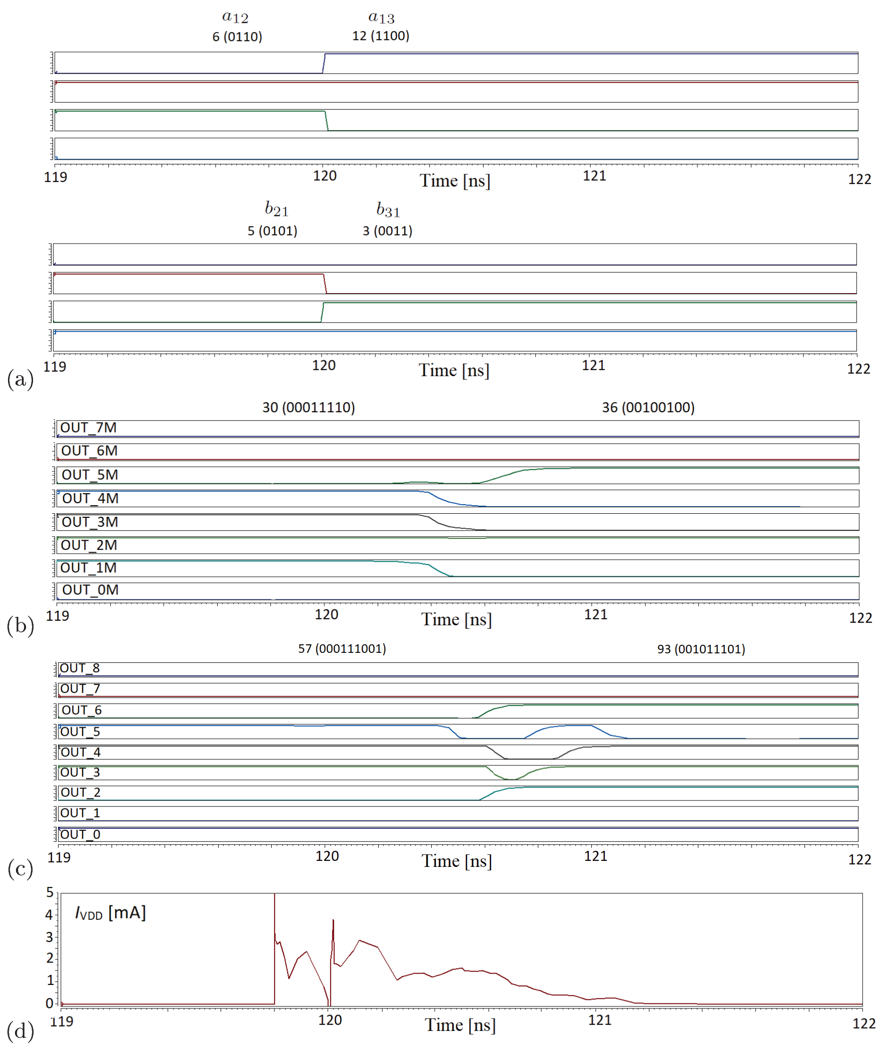

Fig. 4: Zoomed view of a selected computation cycle (a single multiplication-accumulation operation), shown in Fig. 3: (a) selected input signals a_{1m} and b_{n1} from Table 4.1, (b) outputs of the multiplier, (c) outputs of the MBFA, and (d) the supply current.

4.1 Simulation results

To illustrate the performance of the proposed circuit, two exemplary matrices $A_{3,4}$ and $B_{4,2}$ were selected to get the matrix $C_{3,2}$ at the output. Selected elements of those two matrices, required to compute a single element c_{11} of the C matrix are shown in Table 4.1. To facilitate the illustration of

Table 2: Selected input values used in a single MAC circuit

It.	a_{1m}	value	b_{n1}	value	c_{11}
1	a_{11}	3 (0011)	b_{11}	9 (1001)	27
2	a_{12}	6 (0110)	b_{21}	5 (0101)	57
3	a_{13}	12 (1100)	b_{31}	3 (0011)	93
4	a_{14}	8 (1000)	b_{41}	2 (0010)	109

the results, the input signals are 4-bit numbers in this case. Any extension toward higher resolutions requires only increasing the sizes of the MBFAs and the memory block in the MAC.

The following elements of the input matrices were sequentially provided to particular inputs of the matrix multiplication circuit shown in Fig. 1 with data rate up to 100 Msamples/s. Selected simulation results for a single MAC are presented in Fig. 3.

Fig. 3 (a) presents waveforms of the input signals from Table 4.1. Diagram (b) of Fig. 3 shows waveforms of particular bits of the multiplier output, while the diagram (c) presents the computed output signal c_{11}. The energy consumption is assessed on the basis of the supply current, I_{DD}, as shown in the diagram (d), and the supply voltage equals 1.8 V in this case. As can be observed, immediately after providing a new pair of elements a_{1m} and b_{n1}, the I_{DD} current rises and then drops after less than 2 ns. This shows that a single multiplication-accumulation cycle lasts about 2-3 ns. In the parallel approach, in which each element of the C matrix is computed using a separate MAC, the overall multiplication operation for four iterations is completed within 12 ns (the worst case). An average value of I_{DD} equals about 0.8 mA, so the energy consumption per single iteration equals approximately 4.3 pJ. These results are presented for the CMOS 180 nm technology. A substantially shorter computation time is expected in newer processes.

4.2 Laboratory measurement verification

The measurements were carried out in a prototype chip designed in the 130 nm CMOS technology. The chip contains a circuit whose structure corresponds to the structure of a single asynchronous MAC with the memory consisting of NOT gates and switches controlled by a 2-phases clock. Due to the limited number of digital inputs of the chip (11), it was possible to multiply only 4-bit signals (8 inputs). One of the remaining inputs allows for switching between the programming and regular work modes over the chip [30]. The remaining two inputs are used to provide control clock signals to the MAC. It is also possible to reset the ACU block, which is done by setting both clock inputs simultaneously into the logic value of '1'.

The laboratory tests were also performed for the input signals collected in Table 4.1 for the comparison. The measurement results recorded by the MyRio measurement card working under the control of the LabView environment are available in the xls file (after some elaboration). At the level of the logic signals, the circuit works identically to the simulations. For this reason, these results are not presented graphically.

Since the circuit is a part of a larger digital block, it is not possible to demonstrate the measured supply current separately for this block. Another parameter which was also verified in laboratory tests was the minimum data rate needed to avoid visible impact of the leakage phenomenon on the data stored in the memory block. Its value equals 26 ksamples/s. However, the minimal sizes of transistors in the NOT gates were used, which caused the parasitic capacitance C_{GS} to have a minimal value. Therefore, the information storage time can be easily improved, without significant increase of the circuit area by increasing the sizes of the transistors in the NOT gates.

5 Discussion of results

Basing on the obtained results as well as on the details of the structures of particular solutions, more general parameters of these circuits can be estimated, including the data rate and the chip area.

5.1 Circuit complexity and performance analysis

Let us consider an exemplary case of the MAC of 16-bit resolution, and $K = 16$ (range in eq. 9). In the asynchronous approach, the number of transistors in the multiplier equals approximately 9000 and the number of layers in the tree equals 4, while the number of MBFAs equals 15. The resolution (in bits) at the outputs of particular MBFAs varies in between 17 and 32, so the total number of 1BFAs equals approximately 300. A single 1BFA is composed of 28 transistors. To ensure that the output resolution of the multiplier at the level of 16 bits equals the resolution of the input signal, the 16 least significant bits can be omitted. In this case, the MBFA used in the ACU with the resolution of 20 bits and the corresponding memory require additionally about 1000 transistors (10000 in total).

The serial solution requires only one 32-bit MBFA with the 32-bit memory block, so the total number of transistors equals approximately 1300. This number embraces c. 900 transistors in the MBFA and 256 in the memory block realized as a chain of the NOT gates and the switches. In the case of using DFFs in the memory block, the number of transistors equals about 1700. As a result, the complexity of the serial circuit is between 5.8 and 7.7 times smaller than the complexity of the parallel-asynchronous one. A 20 times smaller data rate is the price for this advantage.

In the optimization process of the circuit, the values of particular components of the A and B matrices can be also taken into account. If one of these elements is zero, the multiplication and accumulation can be omitted in order to save the calculation time and the consumed energy.

The accumulation operation has to work in two directions – increment and decrement – as particular elements of the A and B matrices can be negative. Since the signals are coded in the two's complement code, only the summing circuit is used in the ACU.

An analysis of the simulation results presented in the previous section allows to assess the performance of the MAC circuit as well as the overall circuit for the matrix multiplication. The performance includes the data rate and the energy consumption. For an exemplary case of the resolution of 16 bits, the data rate for the parallel and the serial approach, respectively, may be expressed as follows:

$$f_\mathrm{P} = \frac{1}{K \cdot (5 + 20)} \quad [\mathrm{Gsamples/s}] \tag{14}$$

$$f_\mathrm{S} = \frac{1}{K \cdot 16 \cdot 20} \quad [\mathrm{Gsamples/s}] \tag{15}$$

The factor 5 in (14) is the time (in [ns]) which results from the delay of the asynchronous multiplication circuit, while the factor 20 in 14 and 15 is the time (in [ns]) needed for a single accumulation. For the resolution of 16 bits and $K = 16$ $f_\mathrm{P} = 2.5$ Msamples/s, while $f_\mathrm{S} = 0.195$ ksamples/s. This means that the parallel circuit is about 13 times faster than the serial one.

5.2 In the context of existing state-of-the-art solutions

Exemplary realizations of the matrix multiplication operation are provided in Table 2.1. The processed matrices differ significantly in sizes in particular cases (between 4×4 and 512×128). Additionally, they differ in the resolution of the input signals. Taking this into account, different approaches from those presented in section 3 may be applied in each of these cases.

In [21] and [22], for example, the sizes of the matrices allow to apply the parallel approach with 16 MAC circuits. In [22] where the signal resolution equals 4, one can use an asynchronous multiplier with two layers (solution S2.b). In [21] where the signal resolution equals 16, one can use an asynchronous multiplier with four layers (S2.b) or synchronous approach with a single MBFA in the MAC (solution S2.a). In the second approach, the total number of accumulation iterations equals 64 ($4 \cdot 16$). Assuming that the sizes of the matrices are fixed, one can also use the mixed approach (solution S3) described by Eq. 13, with a single MAC, a 2-layer BT at the input and 16 clock iterations.

In [23] 10×10 matrices are processed. In the case of applying the S2.b solution, the number of MACs would equal 100, while the number of the clock iterations – 10. For the signal resolution of 6 bits the minimum number of layers would equal 3. For the memory base on the DFF, the number of transistors in a single MAC would approximately equal $45 \cdot 30$ (BT) $+ 12 \cdot 30$ (MBFA) $+ 12 \cdot 26$ (ACU) ≈ 2000, and 200,000 in the overall circuit (for 100 MACs). Applying the S2.a solution would allow to reduce the number of transistors to about 700 (a single MAC) at the expense of increasing the number of clock iterations to 60 (6 in case S2.b). In this case the mixed approach can also be used. However, the BT at the input would be more complex than in the previous case, described above. In this case, for $K = 10$, the BT with four layers would have to be used.

The most complex situation is the one reported in [24]. Fully parallel approach is rather not optimal, as it would require more than 65,000 MAC circuits. In this situation a compromise between the full parallelism and the circuit complexity is a better option. One of the options could involve applying for example 128 MACs working in parallel, and computing particular columns of the output matrix C in parallel.

5.3 Impact on the overall Kalman filter

Equations 14, 15 and a factor $N_\mathrm{C} \cdot M_\mathrm{C}$ allow the assessment of the overall performance of the Kalman filter. In many applications reported in the literature, the sizes of the matrices do not exceed 4 – 5 [10]. In the tracker used in many AS functions, the sizes of the matrices $\mathbf{A}$, $\mathbf{B}$ as well as the vectors $\mathbf{v}$, $\mathbf{w}$ do not exceed 3×3. In Eqs. (1) and (2) three operations of matrix multiplication are required, so the total number of the MACs (in a fully parallel approach) equals 27. In the case of using the parallel and asynchronous MACs, the calculation time of the overall circuit equals 75 ns, while the total number of transistors equals approximately 300.000 (1.5 mm^2). Even under the worst case conditions (according to process, voltage and temperature variation) this time does not exceed 150 ns. The question is, if the data rate at the level of 6.5 Msamples/s is necessary in this case. A better option might be to apply the serial MAC in this case. For the resolution of 16-bits, the achievable data rate will drop to about 0.5 Msamples/s (in the worst case), which is sufficient in the tracker application. On the other hand, the complexity of the resultant circuit will be much smaller in this case (about 45.000 transistors only).

6 Conclusions

The paper presents selected methods suitable for fully digital implementation of the operation of matrix multiplication. The main objective of this work was to achieve as much parallel signal processing as possible taking into account various trade-offs. The literature study shows that in some applications the sizes of the matrices are very large. In such cases, using mixed solutions which introduce serial data processing at selected computation stages of the overall multiplication operation is a more reasonable approach.

The motivation behind this work was the possibility of implementing fast Kalman filters, in which operations on matrices are commonly performed. In the automotive applications, in many active safety functions, the sizes of the matrices are not very large. This enables an approach in which all elements of the output matrix are calculated in parallel. Simulation as well as measurement tests carried out have shown that it is possible to achieve data rates exceeding 10 Msamples/s, where one sample means full multiplication of two matrices. As data samples (images, radar scans) in the AS functions are provided with rather small rates, considerable amount of computing power remains available. This allows for applying a solution in which a single circuit is multiplexed and shared between various tasks.

The obtained data rates at the level of 2-30 Msamples/s depend on the approach. In comparison to other solutions of this type, average energy consumption at the level of 100 pJ (for 16 bits of the resolution) is promising.

The work presents several practical solutions/approaches. The most interesting are the ones that offer parallel data processing to the highest extent. In many situations, however, various mixed solutions may be optimal, which is also discussed in the paper. One of the effects of the optimization of the core matrix multiplier itself is the reduction of the complexity of the control clock system, which is also an important feature.

References

[1] M. Komorkiewicz, K. Turek, P. Skruch, T. Kryjak and M. Gorgoń, "FPGA-based hardware-in-the-loop environment using video injection concept for camera-based systems in automotive applications", In Proceedings of the 2016 Conference on Design & Architectures for Signal & Image Processing (DASIP). Rennes, France: IEEE, 2016, pp. 183-190.

[2] R. Dlugosz, M. Szulc, M. Kolasa, P. Skruch, K. Kogut, P. Markiewicz, M. Orlowski, M. Rozewicz, A. Ryszka, D. Sasin and T. Talaśka, "Design and optimization of hardware-efficient filters for active safety algorithms", *SAE International Journal of Passenger Cars – Electronic and Electrical Systems*, vol. 8, no.1, pp. 41-50, May 2015.

[3] P. Skruch, M. Dlugosz and W. Mitkowski, "Mathematical methods for verification of microprocessor-based PID controllers for improving their reliability", *Eksploatacja i Niezawodnosc – Maintenance and Reliability*, vol. 17, no. 3, pp. 327-333, 2015.

[4] P. Markiewicz, K. Kogut, M. Różewicz, P. Skruch and R. Starosolski, "Occupancy grid fusion prototyping using automotive virtual validation environment", In Proceedings of the 6th International Conference on Control, Mechatronics and Automation (ICCMA). Tokyo, Japan: IEEE, 2018, pp. 81-85.

[5] M. Komorkiewicz, T. Kryjak, K. Chuchacz-Kowalczyk, P. Skruch and M. Gorgoń, "FPGA based system for real-time structure from motion computation", In Proceedings of the 2015 Conference on Design & Architectures for Signal & Image Processing (DASIP). Kraków, Poland: IEEE, 2015, pp. 1-7.

[6] D. Stepner, N.Rajan and D.Hui, "Embedded Application Design Using a Real-Time OS", In Proceedings of the 36th Design Automation Conference. New Orleans, LA, USA: IEEE, 1999, pp. 151-156.

[7] R.E. Kalman, "A new approach to linear filtering and prediction problems", *Transactions of the ASME Journal of Basic Engineering*, vol. 82, no. 1, pp.35-45, 1960.

[8] S. Gannot, D. Burshtein and E. Weinstein, "Iterative and Sequential Kalman Filter-Based Speech Enhancement Algorithms", *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 4, pp. 373-385, 1998.

[9] G. H. Elkaim, "The Atlantis Project: A GPS-Guided Wing-Sailed Autonomous Catamaran", *Journal of the institute of navigation*, vol. 53, no. 4, pp. 237-247, 2006.

[10] S. Zhao, Y. S. Shmaliy and F. Liu, "Fast Kalman-Like Optimal Unbiased FIR Filtering With Applications", *IEEE Transactions on Signal Processing*, vol. 64, no. 9, pp. 2284-2297, 2016.

[11] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "A Solution to the Simultaneous Localization and Map Building

(SLAM) Problem", *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229-241, 2001.

[12] R. Frohwirth, "Application of Kalman Filtering to Track and Vertex Fitting", *Nuclear Instruments and Methods in Physics Research*, vol. 262, no. 2-3, pp. 444-450, 1987.

[13] B. Shalom and X.R. Li, "Estimation and Tracking: Principles, Techniques and Software", *Artech House Boston.* Boston: Artech House Publishers, 1993.

[14] F.Liu and L.Xia, "Wavelength Assignment of Matrix Multiplication Communication Pattern on a Class of Regular WDM Optical Networks", In Proceedings of the 4th IFIP International Conference on Network and Parallel Computing. Liaoning, China: IEEE, 2007.

[15] Y. Song, R. Jiao, D. Zhang and D. Gao, "Performance Analysis for Matrix-Multiplication Based on an Heterogeneous Multi-core SoC", In Proceedings of the IEEE 11th International Conference on ASIC (ASICON). Chengdu, China: IEEE, 2015.

[16] L.Ni, Y. Wang, H. Yu, W.Yang, C.Weng and J.Zhao, "An Energy – efficient Matrix Multiplication Accelerator by Distributed In-memory Computing on Binary RRAM Crossbar", In Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Macau, China: IEEE, 2016.

[17] R. Frohwirth, "Application of Kalman Filtering to Track and Vertex Fitting", *Nuclear Instruments and Methods in Physics Research*, vol. 262, pp. 444-450, 1987.

[18] R. Długosz, A. Rydlewski and T. Talaśka, "Novel, low power, nonlinear filatation and erosion filters realized in the CMOS Technology", *Facta Universitatis*, vol. 28, no. 2, pp. 237-249, 2015.

[19] K. Venkataraman and T. Sadasivam, "FPGA Implementation of modified ellipitc curve digital signature algorithm", *Facta Universitatis*, vol. 32, no. 1, pp. 129-145, 2019.

[20] A. Napieralski, J. Cłapa, K. Grabowski, M. Napieralska, W. Sankowski, P. Sękalski and M. Zubert, "Image and video proccesing with FPGA support used for biometric as well as others applications", *Facta Universitatis*, vol. 28, no. 2, pp. 165-175, 2015.

[21] R. Lin, "A Reconfigurable Low-power High-Performance Matrix Multiplier Architecture With Borrow Parallel Counters", In Proceedings International Parallel and Distributed Processing Symposium. Nice, France: IEEE, 2003.

[22] P. Saha, A. Banerjee, P. Bhattacharyya and A. Dandapat, "Improved matrix multiplier design for high-speed digital signal processing applications", *IET Circuits, Devices & Systems*, vol. 8, no. 1, pp. 27-37, 2014.

[23] M. Bavandpour, M. R. Mahmoodi and D. B. Strukov, "Energy-Efficient Time-Domain Vector-by-Matrix Multiplier for Neurocomputing and Beyond", *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1-1, 2019

[24] R. Genov and G. Cauwenberghs, “Charge-Mode Parallel Architecture for Vector–Matrix Multiplication”, *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 48, no. 10, pp.930-936, 2001.

[25] S. M. Bozic, *“Digital and Kalman Filtering”*. London, Edward Arnold, 1979.

[26] H.R.Naghizadeh, M.S.Moghadam, S.I.Tous and A.Golmakani, “Design of Two High Performance 1-Bit CMOS Full Adder Cells”, *International Journal of Computing and Digital Systems*, vol. 2, no. 1, pp. 47-52, 2013.

[27] Raushan Kumar, Sahadev Roy, and C.T. Bhunia, “Low-Power High-Speed Double Gate 1-bit Full Adder Cell”, International Journal of Electronics and Telecommunications, vol. 62, no. 4, pp. 329-334, 2016.

[28] S.-M. Kang and Y. Leblebici, *CMOS digital integrated circuits*, The McGraw-Hill Education, 2003.

[29] R. Shalem, E. John, and E. John, “A novel low power energy recovery full adder cell”, In Proceedings of the 9th Great Lakes Symposium on VLSI. Ypsilanti, MI, USA: 1999, pp. 380-383.

[30] M. Banach, T. Talaśka, J. Dalecki and R Długosz, “New Technologies for Smart Cities - High Resolution Air Pollution Maps Based on Intelligent Sensors”, *Concurrency and Computation: Practice and Experience*, 2019.