DOI: https://doi.org/10.15276/aait.06.2023.1
UDC 004.032.26

Methods and hardware to accelerate the work of a convolutional neural network

Ivan G. Tsmots[1)]
ORCID: https://orcid.org/0000-0002-4033-8618; ivan.tsmots@gmail.com. Scopus Author ID: 24484154400
Oleh M. Berezsky[2)]
ORCID: https://orcid.org/0000-0001-9931-4154; olber62@gmail.com. Scopus Author ID: 6505609877
Mykola O. Berezkyy[2)]
ORCID: https://orcid.org/0000-0001-6507-9117; mykolaberezkyy@gmail.com. Scopus Author ID: 58020232600
[1)] Lviv Polytechnic National University, 12, Bandera Str. Lviv, 79000, Ukraine
[2)]West Ukrainian National University, 11, Lvivska Str. Ternopil, 46009, Ukraine

ABSTRACT

Three main approaches to building computer systems are analyzed and allocated: software, hardware, and problem-oriented. A problem-oriented approach was chosen for the implementation of CNN. This approach uses a processor core with hardware accelerators that implement basic CNN operations. The development of computer systems for the implementation of CNN should be carried out based on an integrated approach. This approach includes a modern element base, existing hardware, and software for the implementation of the CNN; methods and algorithms for the implementation of CNN; methods, algorithms, and VLSI structure for the implementation of basic operations of the CNN; methods and means of computer-aided design of hardware and software focused on the implementation of CNN computer systems. For the development of computer systems for the implementation of CNN chosen approach, which includes: variable composition of equipment; use of the basis of elementary arithmetic operations; organization of the process of calculating the scalar product as execution single operation; pipeline and spatial parallelism; localization and simplification of links between the steps of the conveyor; coordination of the time of formation of input data and weighting coefficients with the duration of the conveyor cycle. It is shown that in order to reduce the processing time of large images, it is most expedient to use parallel-stream VLSI -implementation of basic operations. The modified Booth algorithm for forming partial products in a parallel-threaded computing device is selected, which decreased the number of steps in the pipeline. The method of group summation has been improved, which, with multi-input single-digit adders, combined according to the principle of the Wallace tree, provides a reduction in summation time. The method of parallel-flow calculation of scalar product in a sliding window is developed, which, by coordinating the time of receipt of columns of input data and weighting coefficients with the duration of the conveyor cycle, provides high efficiency of equipment use and calculations in real-time. The main ways regarding coordination of the time of receipt of input data columns and weighting coefficients with the duration of the conveyor stroke of hardware that implement two-dimensional convolution are determined. The hardware structure for the realization of two-dimensional convolution in a sliding window, which is focused on VLSI- implementation with high efficiency of equipment use, has been developed. Programmable logic integrated circuits selected for the implementation of hardware accelerators. Single-bit 7, 15, and 31 input adders were developed and modeled on the basis of FPGA EP3C16F484 of the Cyclone III family of Altera company, and an 8-input 7-bit adder was synthesized on their basis.

Keywords: Convolutional neural networks; hardware accelerator; problem-oriented approach; parallel-stream implementation; multi-input adder; scalar product; two-dimensional convolution

For citation*:* Tsmots I. G., Berezsky O. M., Berezkyy M. O. "Methods and hardware to accelerate the work of a convolutional neural network". *Applied Aspects of Information Technology*. 2023; Vol.6 No.1: 13–27. DOI: https://doi.org/10.15276/aait.06.2023.1

INTRODUCTION

Convolutional neural networks (CNN) are a class of deep artificial neural networks of direct spreading, which is focused on effective image recognition and video analysis. The CNN consists of layers of input, output and a number of hidden layers (convolutional, aggregative, fully connected and normalization). The main layer of CNN is the convolution layer, which is the basis of the network [1, 2]. The parameters of the convolution layer consist of filters set for training.

Each filter has small parameters (width and length) and runs through the entire input image. During the passage, each filter slides along the width and length of the input data and calculates the scalar product between the values of the filter parameters and the values of the image parameters. As the filter passes through the width and length of the image, a two-dimensional activation map is compiled, which provides a response of this filter at each spatial position.

Algorithms of CNN can be implemented by software or hardware. Each of the means of implementing CNN has its advantages and disadvantages.

The advantages of software implementation are flexibility in replacing and modifying algorithms, and the main disadvantage is low speed. The advantages of the hardware implementation of the CNN algorithms are high speed, which is achieved due to the specialization and parallelization of processing. Hardware implementation is used in the case when the algorithms are fully worked out and they will not change during operation. The disadvantages of this implementation are the complexity of modifying and changing processing algorithms and high hardware costs. The implementation of CNN using only one of these tools is rare. Mainly for the implementation of CNN used problem-oriented approach, which involves a combination of software and hardware. The process of combining software (universal) and hardware (specialized) provides high efficiency of equipment use and reduces the implementation time of the CNN. With this approach, the implementation of the CNN with the specified technical parameters is reduced to supplementing the processor core with the necessary hardware accelerators that implement basic time list operations of the CNN. Such operations include calculating the scalar product in the sliding filter (two-dimensional convolution) with the window size $m \times m$ pixels.

1. ANALYSIS OF LITERARY DATA AND PROBLEM STATEMENT

Analysis of approaches to the construction of computer systems focused on the implementation of CNN [1, 2], [3] showed that from the set of existing approaches can be distinguished the following:

- the first is a software implementation based on a neurosignal processor, digital signal processing processor, SoC system-on-chip or general-purpose processor;
- the second – hardware implementation in the form of specialized systems, the architecture and organization of the computing process in which reflects the structure of algorithms for the implementation of the CNN;
- the third is a problem-oriented implementation using a processor core supplemented by hardware accelerators that implement basic CNN operations.

The disadvantages of the first approach are low speed, functional and structural redundancy of computer tools [4, 5], [6]. The disadvantage of the second approach is the greater hardware complexity [7, 8]. The third approach provides high efficiency of equipment use and adaptation to the requirements of a particular application [9, 10].

Analysis of the element base in hardware accelerators CNN shows that for their implementation it is advisable to use systems on a chip SoC and programmable logic integrated circuits such as FPGA.

The characteristics of the hardware accelerator for calculating two-dimensional convolution largely depend on approaches to the hardware implementation of the scalar product calculation operation. Analysis of work [2, 5] shows that there are two approaches to the hardware implementation of the operation calculating scalar products. The first of them is based on multiplication and addition operations, and the second is based on elementary arithmetic operations of addition, inversion and shift. The first approach is mainly used to calculate the scalar product as a set of multiplication and addition operations. This approach does not provide optimization of the device structure and its time parameters. Using the basis of elementary arithmetic operations and the multi-operation approach optimizes structure of device in terms of speed and hardware costs. The basis of algorithms for calculating the scalar product using the multi-operand approach and the basis of elementary arithmetic operations is the formation of partial products with their subsequent addition.

The analysis of device structures [5, 12], [13], which are used to implement algorithms for calculating scalar and basis elementary arithmetic operations showed that two types of structures are used for implementation: recursive and non-recursive. A structural feature of recursive devices is the presence of inverse relationships. In such devices, the calculation of the scalar product is carried out in several iterations, the number of which is determined by the algorithm for forming partial products. The disadvantage of recursive devices for calculating the scalar product is relatively low speed.

Non-recursive devices have greater speed, a feature of which is the absence of inverse connections.

Such devices are divided into two classes:

– the first is matrix, which uses parallel formation and summation of all partial products;

– the parallel-thread approach is parallel-streaming, which uses sequential formation and addition with the corresponding offset of partial products.

The disadvantage of matrix devices for calculating the scalar product is the heterogeneity of the structure. Parallel-stream devices for calculating the scalar product have a homogeneous structure

with regular couplings and are more focused on VLSI implementation.

From the analysis of the literature [5, 14], [15] it follows that the speed of parallel-stream devices for calculating the scalar product largely depends on the speed of implementation of group summation of operands.

In [5, 16], [17] the horizontal model of group summation, which has a relatively low speed, is considered.

In articles [18, 19], [20-27] hardware accelerators for CNN based on FPGAs. The problem of improving FPGA-components developed for critical application systems is discussed in articles [28, 29]. Applied aspects of the application of deep neural networks are analyzed in [30].

Consequently, the analysis of literature data showed that the development of highly efficient hardware accelerators for the implementation of two-dimensional convolution requires the improvement of the method of group summation, the development of a new method and structure of parallel-flow calculation of the scalar product in a sliding window.

Therefore, the actual problem is the development of a problem-oriented computer system for the effective implementation of the CNN main operations.

GOAL AND RESEARCH OBJECTIVES

The purpose of the research. The purpose of the research is to develop a method, algorithms and structures to increase the efficiency of equipment use in the hardware implementation of two-dimensional convolution in a sliding window.

The objectives of the research are as follows: analysis of literary data; principles selection of construction and development of the computer system basic structure focused on the implementation of CNN; development of methods and algorithms for the implementation of two-dimensional convolution for the hardware accelerator of the CNN; development of the structure of a parallel-flow device for calculating the scalar product; development of the structure of hardware for the implementation of two-dimensional convolution; implementation of a multi-input adder on FPGA.

The object of research is the processes of parallel-stream calculation of two-dimensional convolution in a sliding window.

The subject of research are methods, algorithms and structures of hardware components for calculating two-dimensional convolution in a sliding window.

2. MAIN RESEARCH RESULTS

2.1. Basic structure of a computer system based on an integrated approach

The development of computer systems focused on the implementation of CNN was carried out based on the integrated approach, which covers:

- modern element base, existing hardware and software for the implementation of CNN;
- methods and algorithms for the implementation of CNN;
- methods, algorithms and VLSI structures for the implementation of basic CNN operations;
- methods and tools for computer-aided design of hardware and software focused on the implementation of CNN computer systems.

The development of computer systems and hardware accelerators for the implementation of complex basic operations of the CNN is based on [5]:

- variable composition of equipment, which provides for the presence of a processor core and replaceable hardware modules that implement basic CNN operations with great computational complexity;
- modularity, which involves the development of hardware accelerators in the form of modules that have access to a standard interface;
- use of the basis of elementary arithmetic operations for VLSI-implementation of basic operations of CNN;
- organization of the calculating process in scalar product as a single operation;
- pipeline and spatial parallelism in the implementation of CNN and basic operations;
- of the conveyor steps in the implementation of the scalar product;
- localization and simplification of links between conveyor steps;
- ensuring a balance between input-output and calculations;
- minimization of the external communication interface.

The basic structure of a computer system focused on the implementation of CNN is presented in the form of a constant particle – processor core and variable part – hardware accelerators that implement complex basic operations of the CNN [3, 5]. The basic structure of a computer system focused on the implementation of CNN is shown in Fig. 1.

The symbols are as follows: MPM – multiport memory; BO is a basic operation.

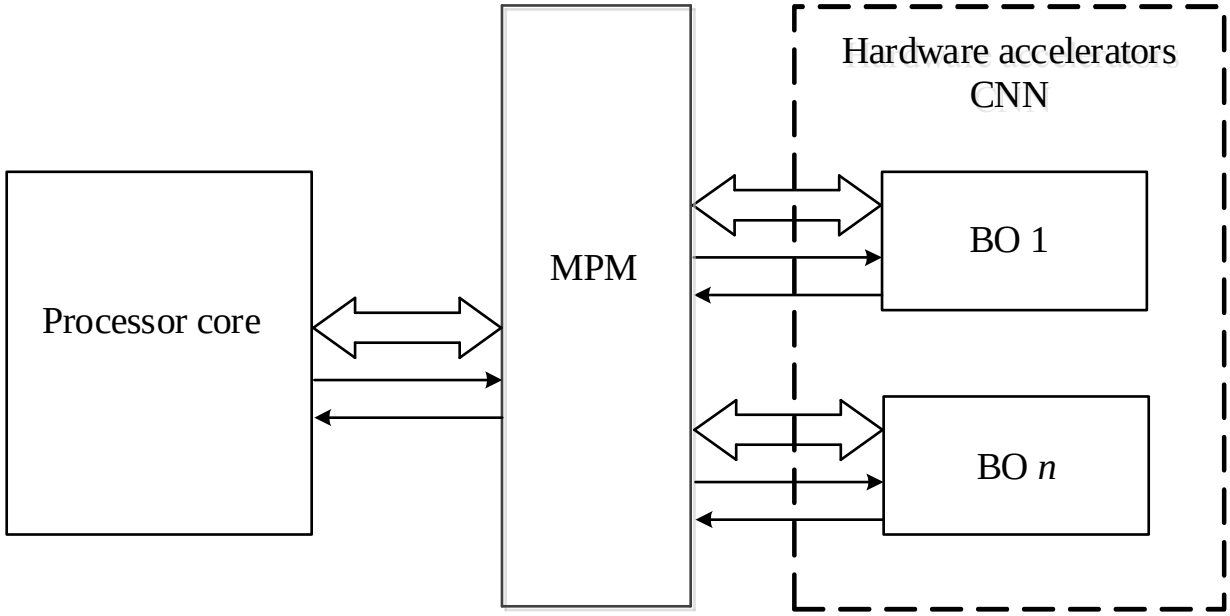

Fig. 1. **The basic structure of a computer system focused on the implementation of CNN**
Source: **compiled by the authors**

The main components of a computer system are the processor core, a set of hardware accelerators that hardware-implement control algorithms for BO implementing CNN and MPM.

The processor core is designed to control the computational process and implement parts of the CNN algorithms in which logical operations and heterogeneous calculations prevail. Such a core can be implemented on the basis of a neural signal processor, digital signal processing processor, system-on-chip SoC or general-purpose processor.

In a computer system, MPM is used to reduce time and synchronize the exchange between the processor core and hardware accelerators. The use of MPM provides: conflict-free data exchange, high speed transfer of data, the ability to change the width and time of memory access, work with different speed hardware accelerators and the ability to simultaneously connect the necessary hardware accelerators. The peculiarity of MPM is independent sets of address and data that come from the processor core and hardware accelerators.

2.2. Method and algorithms of two-dimensional convolution for the hardware accelerator CNN

The initial information for the development of a hardware accelerator to perform the operation of calculating a two-dimensional convolution is:

- image size;
- the size of the filter window;
- intensity of input data and weights;
- interface requirements;
- bit depth of input data, weights and accuracy of calculations;
- technical and economic requirements and restrictions.

In parallel-flow structures, data is processed according to the pipeline principle [5]. Pipelining involves dividing structures into steps, each of which consists of two blocks – operational and buffer memory. Management of parallel-stream means is reduced to the issuance of clock pulses that move data from input to output, writing intermediate results to buffer memory. The frequency of receipt of clock pulses is determined by the time of melting to the buffer memory and the delay time at the operating unit.

In parallel-stream calculation of two-dimensional convolution in a sliding filter with a window of $m \times m$ pixels, the input data (pixels) X_{kj} and the weights W_{kj} ($j = 1, ..., m$; $k = 1, ..., m$) come simultaneously to all inputs in parallel binary code with a fixed point.

The calculation of such a two-dimensional convolution reduces to the calculation of the scalar product for the j^{th} input column:

$$Z_k = \sum_{j=1}^{m} W_{kj} X_{kj}\,, \qquad (1)$$

and summing in the sliding window m the results of calculating the scalar product according to the formula:

$$Y = \sum_{k=1}^{m} Z_k \ . \qquad (2)$$

To implement a hardware accelerator, it is necessary to develop algorithms for parallel-stream calculation of the scalar product and group summation.

2.2.1. Method and algorithms of parallel-stream calculation of scalar product

For a parallel-stream VLS implementation, the parallel-thread calculation algorithm of the scalar product must be structured, recursive, and locally dependent.

The parallel-stream calculation of the scalar product must be performed on the basis of operations of the same type, which reduces to the formation of macroparticle products P_{Mh} and their addition to previously accumulated amounts in accordance with the formula:

$$Z_h = 2^{-h} Z_{h-1} + P_{Mh}, \qquad (3)$$

where Z_0=0.

The operation of forming the macro partial product P_{Mh} and its addition according to the formula (3) are realized *by h* step of the pipeline. The number of steps of the pipeline is determined by the bit depth of the operands n and the number of digits k, which are analyzed to form macroparticle products P_{Mh}.

The formation of the h^{th} macro partial product P_{Mh} (h=1, …, r, where $r = \left\lceil \frac{n}{k} \right\rceil$, $\lceil\ \rceil$ is the sign of rounding to a larger integer) is performed by summing the group partial products, which are obtained by analyzing k digits in accordance with the formula:

$$P_{Mh} = \sum_{j=1}^{m} (W_j X_{jh1} + 2^{-1} W_j X_{jh2} + \ldots + 2^{-(k-1)} W_j X_{jhk}) = \sum_{j=1}^{m} P_{jh} \ . \qquad (4)$$

For parallel-stream calculation of the scalar product, it is advisable to use algorithms for analyzing lower digits to form partial products. The structure of the pipeline step depends on the number of analyzed bits k and algorithms for forming group partial products P_{jh}. Most often, a modified Booth algorithm is used to form partial products. Calculating the scalar product using the modified Booth algorithm to form partial products involves partitioning factors X_j on groups of three digits, so that neighboring groups have one common digit [5,17]. The division of factors X_j into groups is carried out from the lowest digits, and the lowest digit of the youngest group is always supplemented by zero.

For each q group of bits (q=1, ..., r, $r = \left\lceil \frac{n}{k} \right\rceil$) of factors $X_{j[2(r-q+1)-1]}$ $X_{j[2(r-q+1)]}$ $X_{j[2(r-q+1)+1]}$, the partial product P_{jq} is formed according to the formula:

$$P_{jq} = K_{jq} W_j. \qquad (5)$$

The value of K_{jq} is defined as the sum of the weights of nonzero digits of the factor bit group $X_{j[2(r-q+1)-1]}$ $X_{j[2(r-q+1)]}$ $X_{j[2(r-q+1)+1]}$, where $X_{j[2(r-q+1)-1]}$ has weight minus two, and $X_{j[2(r-q+1)]}$ and $X_{j[2(r-q+1)+1]}$ – unit, according to the expression:

$$K_{jq} = \begin{cases} 2, \ \text{if } X_{j[2(r-q+1)-1]} = 0, X_{j[2(r-q+1)]} = X_{j[2(r-q+1)+1]} = 1, \\ 1, \ \text{if } X_{j[2(r-q+1)-1]} = 0, X_{j[2(r-q+1)]} \neq X_{j[2(r-q+1)+1]}, \\ 0, \ \text{if } X_{j[2(r-q+1)-1]} = X_{j[2(r-q+1)]} = X_{j[2(r-q+1)+1]}, \\ -1, \ \text{if } X_{j[2(r-q+1)-1]} = 1, X_{j[2(r-q+1)]} \neq X_{j[2(r-q+1)+1]}, \\ -2, \ \text{if } X_{j[2(r-q+1)-1]} = 1, X_{j[2(r-q+1)]} = X_{j[2(r-q+1)+1]} = 0. \end{cases} \qquad (6)$$

When forming partial products P_{jq}, the multiplication operation by two is realized by shifting one digit to the left and changing the sign by inversion of all digits of the multiplied followed by adding one to the lowest digit.

After forming the partial products P_{jq}, the q^{th} macropartial product is calculated using the following formula:

$$P_{Mq} = \sum_{j=1}^{m} P_{jq} \ . \qquad (7)$$

The calculation of the scalar product Z with the formation of partial products according to the modified Booth algorithm is performed according to the expression:

$$Z_q = 2^{-2} Z_{q-1} + P_{Mq} \ . \qquad (8)$$

From the formulas (5-8) it can be seen that the algorithms for calculating the scalar product are reduced to performing the same type of operations - the formation of partial products P_{jq}, calculating the q^{th} macropartial product P_{Mq} and adding to the previously accumulated sums shifted by two digits to the right.

2.2.2. Group summation algorithm

To implement the group summation operation, both horizontal and vertical models can be used.

The horizontal model of group summation is implemented by the formula:

$$Y = \sum_{k=1}^{m} \sum_{i=1}^{n} 2^{-i} Z_{ki} , \qquad (9)$$

where n is the bit depth of terms.

In [5, 17] all possible options for implementing the horizontal model of group summation are considered. The fastest version of the implementation of the horizontal model of group summation is a parallel-parallel summation method, the graph of the algorithm of which is shown in Fig. 2.

The parallel-parallel group summation algorithm is cascading. The time for calculating the sum of the macro-operation of group summation according to this algorithm depends on the height of the graph (number of tiers), which is calculated as follows:

$$h = \lceil \log_2 m \rceil . \qquad (10)$$

In each tier, the operands are divided into pairs, for each of which a sum is calculated.

The total number of addition operations to sum m numbers is equal to:

$$U = \frac{m}{2} + \frac{m}{4} + \frac{m}{8} + \cdots + 1 = m - 1 . \qquad (11)$$

The summation time can be reduced by using the vertical and multi-operand addition algorithm [17].

Replacing the summation order in formula (9), we proceed to the vertical model of group summation, which is written as follows:

$$Y = \sum_{i=1}^{n} 2^{-i} \sum_{k=1}^{m_i} Z_{ki} , \qquad (12)$$

where m_i is the number of terms in the i^{th} bit section.

Existing vertical methods of group summation reduce the summation process to converting a multi-row code to a single-row code. This transformation is based on the basic operation of converting a three-line code to a two-row code, which is carried out using a layer of single-digit adders that have no connections with each other. To reduce the conversion time of a multi-row code into a single-row layer, single-bit adders must be combined according to the principle of the Wallace tree.

The number of single-bit adder layers to calculate the group summation operator is determined by the formula:

$$K = \lceil \log_{1.5} 0{,}5m \rceil . \qquad (13)$$

Group summation using this method is considered as performing a single operation where hyphenation units are counted only once at the stage of converting a two-line code to a single-line code.

Acceleration of the converting process of a multi-row code to a single-row code is carried out by using the conversion of 3-, 7- and 15-input single-bit adders.

The work of such adders is described by the following expressions:

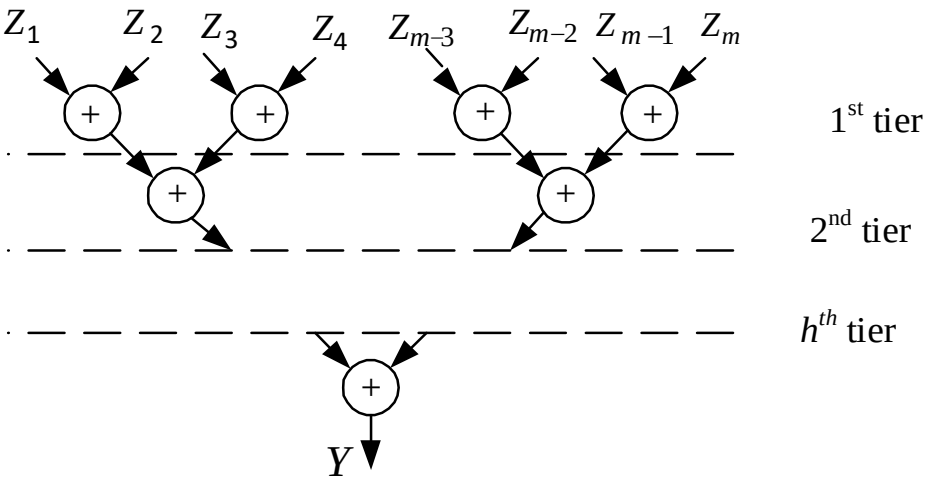

Fig. 2. **Graph of parallel group summation algorithm**
Source: **compiled by the authors**

$$E_{3-2}=\begin{cases}C_{ji}\\+\\C_{(j+1)i}\\+\\C_{(j+2)i}\end{cases}=\begin{cases}P_{i-1}\\+\\S_i\end{cases},\quad E_{7-3}=\begin{cases}C_{ji}\\+\\C_{(j+1)i}\\+\\C_{(j+2)i}\\+\\C_{(j+3)i}\\+\\C_{(j+4)i}\\+\\C_{(j+5)i}\\+\\C_{(j+6)i}\end{cases}=\begin{cases}P_{i-2}\\+\\S_{i-1}\\+\\S_i\end{cases},\quad E_{15-4}=\begin{cases}C_{ji}\\+\\C_{(j+1)i}\\+\\C_{(j+2)i}\\+\\C_{(j+3)i}\\+\\C_{(j+4)i}\\+\\C_{(j+5)i}\\+\\C_{(j+6)i}\\+\\C_{(j+7)i}\\+\\C_{(j+8)i}\\+\\C_{(j+9)i}\\+\\C_{(j+10)i}\\+\\C_{(j+11)i}\\+\\C_{(j+12)i}\\+\\C_{(j+13)i}\\+\\C_{(j+14)i}\end{cases}=\begin{cases}P_{i-3}\\+\\S_{i-2}\\+\\S_{i-1}\\+\\S_i\end{cases},$$

where E_{3-2}, E_{7-3} *and* E_{15-4} are outputs of sums, respectively, on three, seven and fifteen input single-digit adders, C_{ji} is the i[th] digit of the j[th] operand, P_{i-1} is the output of the (i-1)[th] digit of the transfer, S_i is the output of the i[th] digit of the sum. To convert a multi-row code to a two-row code, the union of such adders is used according to the principle of the Wallace tree. The conversion of a two-line code to a single-line code is performed using a parallel adder.

2.3. The structure of the parallel-flow device for calculating the scalar product with the formation of partial products according to the modified Booth algorithm

The structure of a parallel-thread device for calculating a scalar product with the formation of partial products according to the modified Booth algorithm should be focused on the VLSI implementation, involving the use of steps of the same type with local and regular connections. When developing the structure of the device, it is necessary to ensure a balance between I / O and calculations.

The developed structure of the device of parallel-stream calculation of scalar product with formation of partial products according to the modified Booth algorithm is shown in Fig. 3.

For this, the following notation is introduced:
SC – step conveyor; Rg – register; Add – adder; *m*Ad – *m*-input adder; Dec – decoder; BFPP – block of formation of partial products; Sw – switch [17].

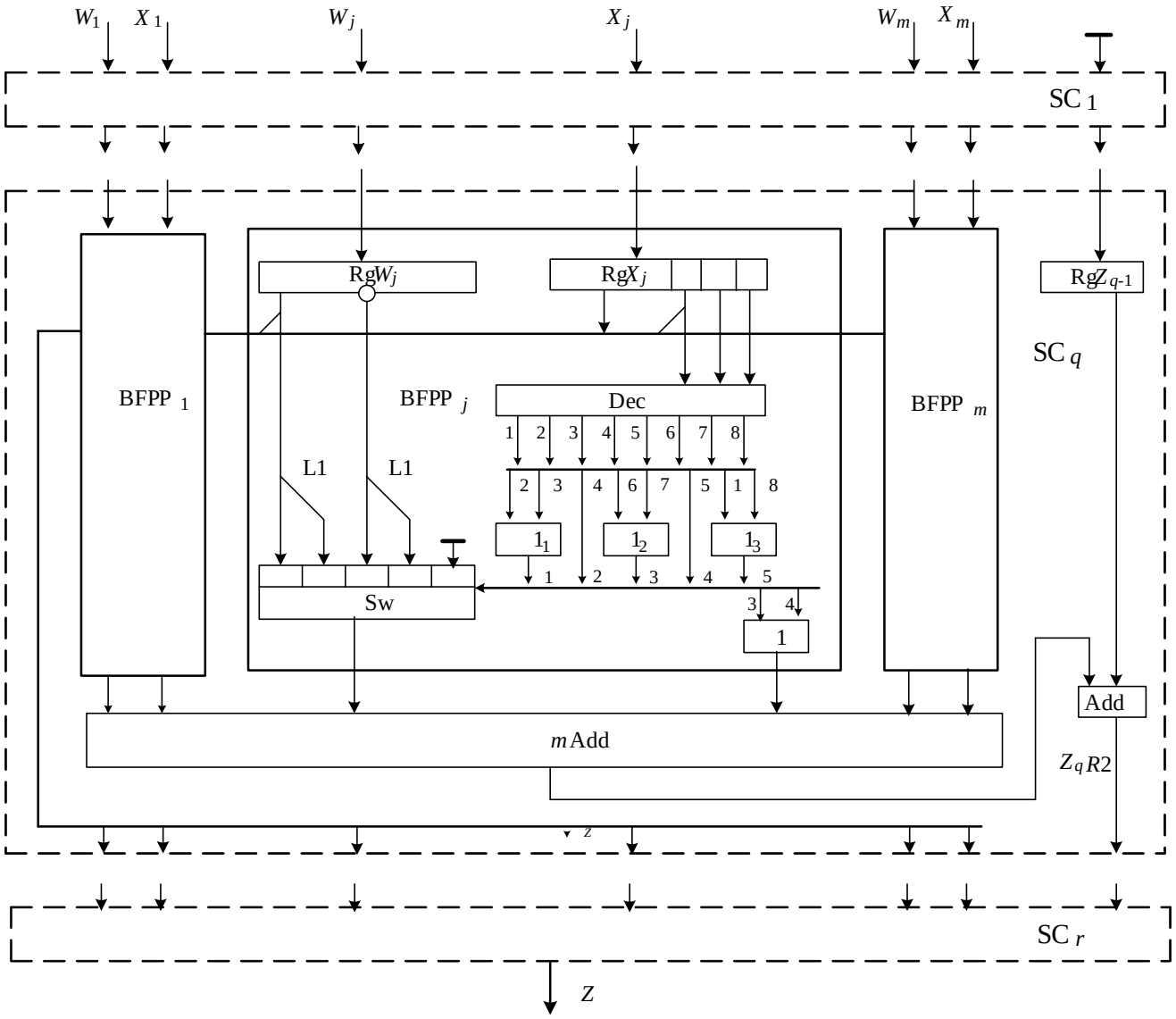

Fig. 3. **The structure of the device for parallel-stream calculation of the scalar product with the formation of partial products according to the modified Booth algorithm**

Source: **compiled by the authors**

The number of steps of the pipeline required for the synthesis of a parallel-flow device for calculating the scalar product is $r = \left\lceil \frac{n}{2} \right\rceil$. In each step of the conveyor SK_q the formation of partial products P_{jq} is performed by the partial product formation block using the analysis of three digits $X_{j[2(r-q+1)-1]}$ $X_{j[2(r-q+1)]}$ $X_{j[2(r-q+1)+1]}$.. Bits $X_{j[2(r-q+1)-1]}$ $X_{j[2(r-q+1)]}$ $X_{j[2(r-q+1)+1]}$ go to the inputs of the decoder. Using logic elements OR and signals from the outputs of the decoder, the formation of control signals of the switch – 10000 ($X_{j[2(r-q+1)-1]}$=0, $X_{j[2(r-q+1)]} \neq X_{j[2(r-q+1)+1]}$), 01000 ($X_{j[2(r-q+1)-1]}$=0, $X_{j[2(r-q+1)]}$=$X_{j[2(r-q+1)+1]}$ =1, 00100 ($X_{j[2(r-q+1)-1]}$=1, $X_{j[2(r-q+1)]} \neq X_{j[2(r-q+1)+1]}$), 00010 ($X_{j[2(r-q+1)-1]}$=1, $X_{j[2(r-q+1)]}$=$X_{j[2(r-q+1)+1]}$=0), i 00001 ($X_{j[2(r-q+1)-1]}$=$X_{j[2(r-q+1)]}$=$X_{j[2(r-q+1)+1]}$). The switch, depending on the signals that come to the control inputs, is installed in the appropriate positions. When its output receives the value W_j (control signal – 10000), value $2W_j$ (control signal – 01000), value ($-W_j$) (control signal – 00100), value ($-2W_j$) (control signal – 00010), logical zero value (control signal – 00001). At the output of the fourth element OR a logical zero signal is generated (with control signals 10000, 01000, 00001) or logical unit (with control signals 00100 and 00010). The formed partial products are fed to the inputs of the *N*-input adder, at the output of which we obtain the macro-component product P_{Mq}. The calculated macro-master product P_{Mq} is added to the partial result Z_{q-1} shifted two digits to the right.

The result of calculating the first scalar product is obtained at the output of the device after the r^{th}

cycle. In each subsequent cycle of work at the output of the device, we will obtain the results of calculating the following scalar products.

This device works with a beat, the duration of which is calculated by the formula:

$$T_{SP_C} = t_{Rg} + t_{Dec} + t_{OR} + t_{Sw} + t_{mAdd} + t_{Add}, \quad (14)$$

where t_{Rg} is time of writing to the register; t_{Dec} is operation time of the decoder; t_{OR} is delay time on the logical element OR; t_{Sw} is delay time of the switch; t_{mAdd} is summation time m macroparticle products; t_{Add} is time of addition of two numbers.

The costs of equipment for the implementation of this device are determined by the expression:

$$W_{SP} = r[N(2W_{Rg} + W_{Sw} + W_{Dec} + 4W_{OR}) + W_{mAdd} + W_{Add} + W_{Rg}], \quad (15)$$

where W_{Rg}, W_{Sw}, W_{OR}, W_{mAdd}, W_{Add}, W_{Dec} are equipment costs, respectively, for the register, switch, logical element *OR; m* – input adder, adder and decoder.

2.4. The structure of hardware for the implementation of two-dimensional convolution

The development of hardware for the implementation of two-dimensional convolution will be carried out on the basis of a device for parallel-stream calculation of the scalar product with the formation of partial products according to the modified Booth algorithm. This algorithm is complemented by a data format converter and a summation block for scalar products in a sliding window. The developed structure of hardware for the implementation of two-dimensional convolution is shown in Fig. 4.

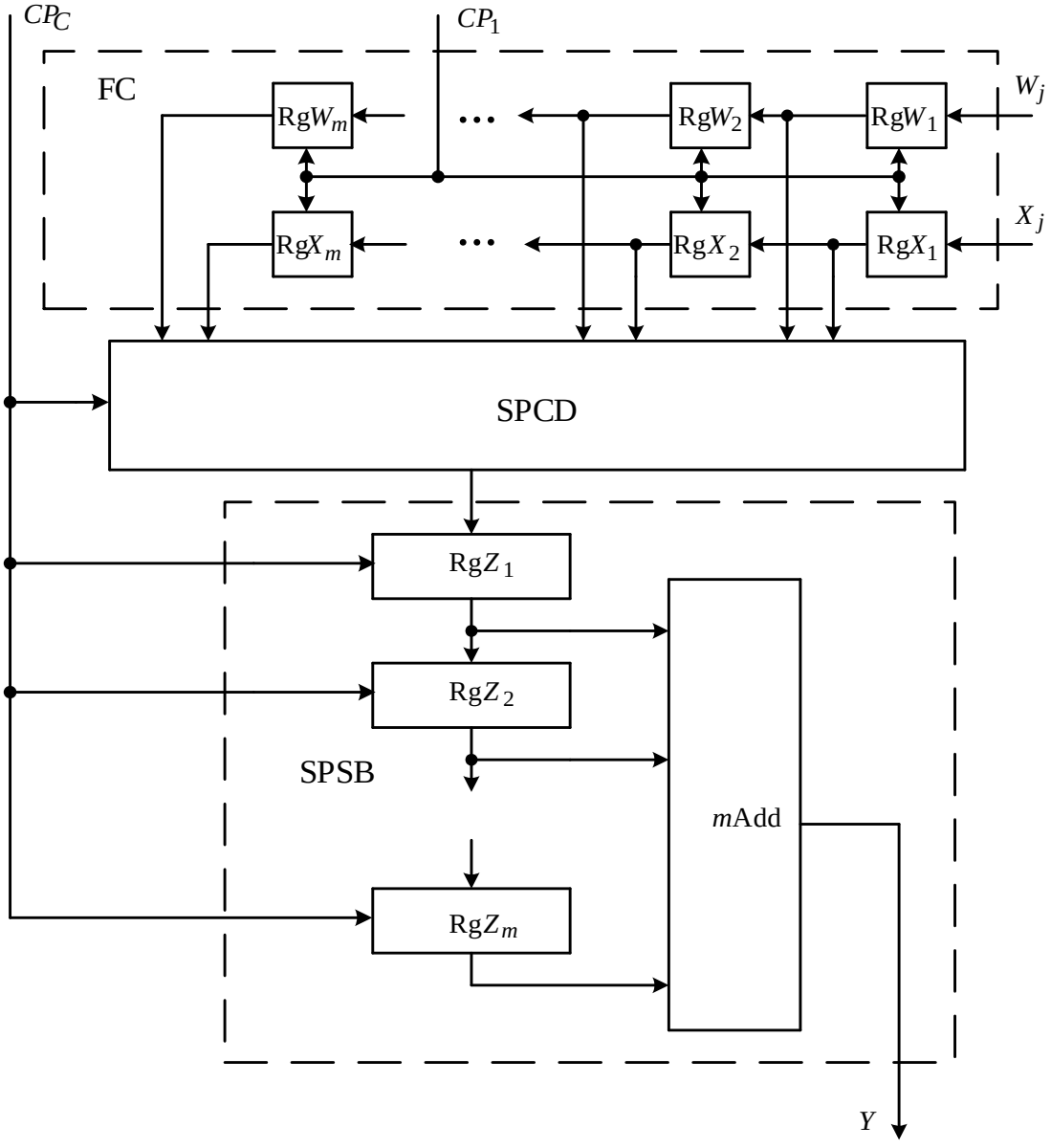

Fig. 4. **The structure of hardware for the implementation of two-dimensional convolution**

Source: **compiled by the authors**

The symbols are as follows: CP_1 is first clock pulses; CP_C is clock pulses of the conveyor; FC format converter; SPCD is a device for calculating the scalar product; SPSB is summation block of scalar products.

The developed hardware is focused on the pipelined parallel-flow implementation of two-dimensional convolution. In parallel-stream calculation of two-dimensional convolution, each clock pulse CP_1 in registers RgX_1 and RgW_1 of the FC format converter are written, respectively, the input data (pixel) X_j and the weight factor W_j. After m clock pulses CP_1 in registers RgX_1, ..., RgX_m and RgW_1, ..., RgW_m are written respectively columns of input data and weighting coefficients. The accumulation time of the column of input data and weights in the FC equal to $t_c = mt_{CP_1}$, where t_{CP_1}, is the period of clock pulses. Controlling the process of calculating two-dimensional convolution Y is reduced to the issuance of conveyor clock pulses CP_1 that advance data from the SPCD input to output Y, writing intermediate results to buffer memory (registers).

To ensure high efficiency of equipment use, all steps of the conveyor must perform approximately the same complexity of operations, and the time of entering the column of input data and weighting coefficients, the stroke of the conveyor and the output time of the results must be coordinated. The number of steps of the pipeline depends on the following parameters: bit depth of input data; the number of digits that are analyzed to form macroparticle products; the size of the sliding window.

The results of calculating two-dimensional convolution after the initial delay are obtained at the output Y in each pipeline cycle.

For effective hardware implementation of two-dimensional convolution, it is necessary to ensure that the following conditions are met:

$$t_c \geq T_C, \quad (16)$$

where T_C is duration of the conveyor cycle.

Conveyor cycle T_C the operation of the hardware for calculating two-dimensional convolution is determined by the beat T_{SP_C} of the work of the SPCD (14). The duration of the conveyor cycle of the work T_C mainly depends on the speed of the element base, the time t_{FPP} of partial products formation and the time t_{mAdd} summation of partial products that determine the complexity of operations in the pipeline step and their number. The main ways to reduce T_C is to conveyor a multi-input adder by breaking it into steps.

The choice of the variant of the hardware accelerator for calculating two-dimensional convolution in a sliding window is carried out according to the criterion of efficiency of use of equipment E_{UE}, which considers the number of interface pins, homogeneity of structure, number and locality of connections. The efficiency criterion links performance with equipment costs and evaluates the elements (valves) of the component by performance [5, 15].

The efficiency of equipment use is determined by the formula:

$$E_{UE} = \frac{R_{TC}}{t_{TC}[(W_{SPCD} + W_{FC} + W_{SPSB})(k_1 + k_2) + W_{Int}k_3]}, \quad (17)$$

where R_{TC} is complexity of algorithms for calculating two-dimensional convolution, which is determined by the number of elementary arithmetic operations necessary for its implementation; t_{TC} is time of calculation of two-dimensional convolution; W_{SPCD} is equipment consumption on the device for calculating the scalar product; W_{FC} is equipment costs for the implementation of the format converter; W_{SPSB} is equipment costs for the implementation of the scalar product summation unit; W_{Int} is hardware costs for interface implementation; k_1 is coefficient of taking into account the homogeneity of the structure; k_2 is coefficient of taking into account the number and locality of links; k_3 is coefficient of taking into account the number of interface outputs.

The cost of equipment for the hardware implementation of two-dimensional convolution is determined as follows:

$$W_{TC} = W_{FC} + W_{SPCD} + mW_{Rg} + W_{m\,Add}, \quad (18)$$

where W_{FC}, W_{SPCD}, W_{mAdd}, W_{Rg} are equipment costs, respectively, on the format converter, the device for calculating the scalar product; m-input combiner and register.

2.5. Multi-input adder on FPGA

The most complex component of the developed hardware for implementing two-dimensional convolution is a multi-input adder. Based on the FPGA EP3C16F484 family of Cyclone III family of Altera company, single-bit 7, 15 and 31 input adders have been developed.

Table 1 shows the hardware resources of FPGAs, which are required to implement single-bit 7, 15 and 31 input adders.

Table 1. **FPGA hardware resources for implementing single-bit multi-input adders**

Number of adder inputs	Number of logical elements	Number of pins
7	20/15408	12/347
15	106/15408	21/347
31	369/15408	38/347

Source: **compiled by the authors**

Based on the developed single-bit multi-input adders, an eight input seven-bit adder MSm was synthesized. Summation in such a multi-input MSm adder is performed using vertical and multi-operand approaches. Process summation is considered as the execution of a single operation based on multi-input single-digit addition operations.

The summation of numbers in an 8-input 7-bit adder is performed in four stages by using the following transformations: 7-row code to 3-row, 3-row code to 2-line code, and 2-row code to 1-row code. Multiinput adder MSm is shown in Fig. 5.

The interface of the 8-in 7-bit adder consists of the Clk sync input, Reset adder reset input, *W_X1* inputs, ..., *W_X8* and Out_Sm output(10...0). Operands (*W_X1*, ..., *W_X8*), Clk synchronization pulses and Reset adder reset pulse are fed to the input of the multi-input MSm adder. At the output of the multi-input adder MSm through two synchronization pulses Clk we get the sum Out_Sm. The multi-input adder MSm is implemented on the basis of eight single-bit multi-input adders Add_7_3_Sym, sixteen Add_3_2 and eight-bit parallel Add_Bin_Paral adder.

Time diagrams illustrating the operation of the MSm multi-pin adder are shown on Fig. 6.

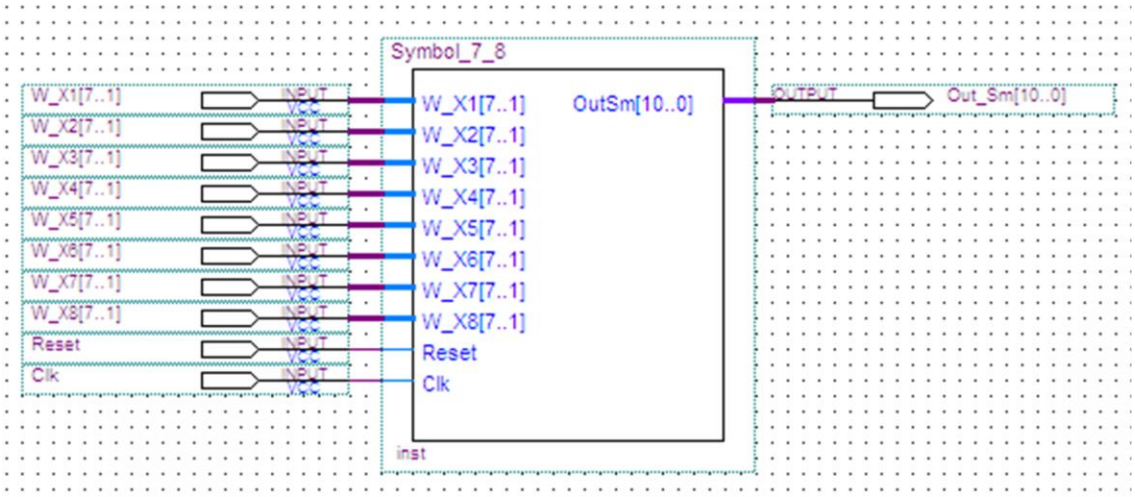

Fig. 5. **Multiple input adder MSm**

Source: **compiled by the authors**

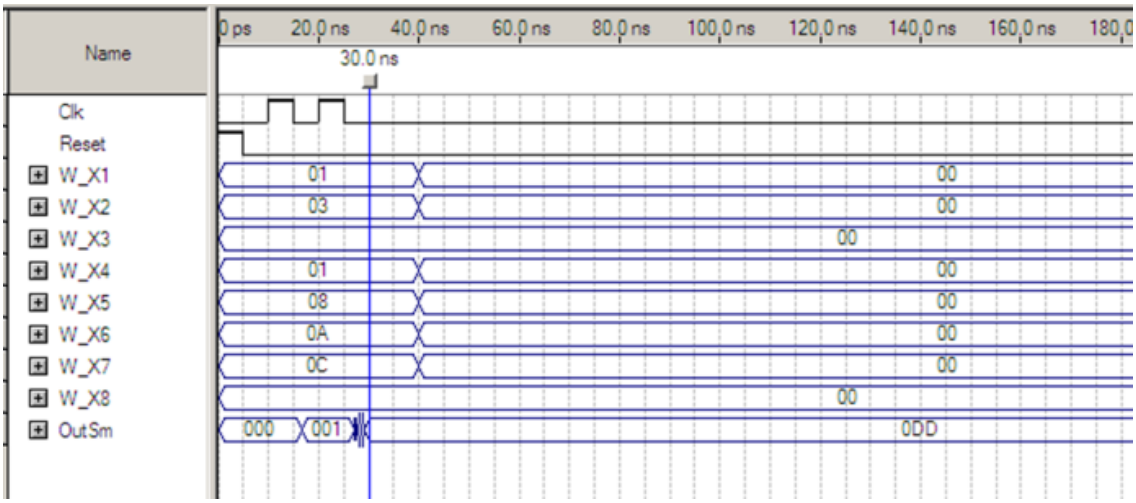

Fig. 6. **Time diagrams of the multi-input adder MSm operation**

Source: **compiled by the authors**

Fig. 6 shows an example of calculating the q[th] macroparticle product by P_{Mq} summation of the partial products of P_{jq} on an 8-input 7-bit adder. For weight coefficients W_1=0001011=B_{16}, W_2=0100010=22_{16}, W_3=0100000=20_{16}, W_4=0111000=38_{16}, W_5=0111010=$3A_{16}$, W_6=0011101=$1D_{16}$, W_7=0101110=$2E_{16}$, W_8=0101100=$2C_{16}$ and values of q groups of bits of input data $K_{1q}=X_{1[2(r-q+1)-1]}\ X_{1[2(r-q)]}\ X_{1[2(r-q+1)+1]}$=001=1; $K_{2q}=X_{2[2(r-q+1)-1]}\ X_{2[2(r-q)]}\ X_{2[2(r-q+1)+1]}$=010=1, $K_{3q}=X_{3[2(r-q+1)-1]}\ X_{3[2(r-q)]}\ X_{3[2(r-q+1)+1]}$=011=2, $K_{4q}=X_{4[2(r-q+1)-1]}\ X_{4[2(r-q)]}\ X_{4[2(r-q+1)+1]}$=001=1, $K_{5q}=X_{1[2(r-q+1)-1]}\ X_{5[2(r-q)]}\ X_{5[2(r-q+1)+1]}$=000=0, $K_{6q}=X_{1[2(r-q+1)-1]}\ X_{6[2(r-q)]}\ X_{6[2(r-q+1)+1]}$=111=0, $K_{7q}=X_{1[2(r-q+1)-1]}\ X_{7[2(r-q)]}\ X_{7[2(r-q+1)+1]}$=000=0, $K_{8q}=X_{1[2(r-q+1)-1]}\ X_{8[2(r-q)]}\ X_{8[2(r-q+1)+1]}$=111=0 the following values are formed partial products P_{1q}=B_{16}, P_{2q}=22_{16}, P_{3q}=40_{16}, P_{4q}=70_{16}, P_{5q}=0_{16}, P_{6q}=0_{16}, P_{7q}=0_{16}, P_{8q}=0_{16}. With the help of an 8-input 7-bit adder, we sum up the partial products P_{1q}, …, P_{8q} and we get at the output *Out_Sm* the macroparticle product P_{Mq}, which is equal to 0*DD*16.

The summation time in an eight-way eight-bit cascade parallel-parallel adder is equal to

$$t_{8\text{AddCPP}}=\log_2 m\times 7\log_2 n\times t_{\log.\text{AND}}=3\times 21\times t_{\log.\text{AND}}=$$
$$=63t_{\log.\text{AND}},$$

and the summation time in an eight-input eight-bit adder using a vertical and multi-operand approaches and 3- and 7-input single-bit adders is equal to,

$$t_{8AddV}=t_{Add_{7-3}}+2t_{Add_{3-2}}+t_{Add_8}=5t_{\log.AND}+$$
$$+6t_{\log.AND}+7\log_2 8\times t_{\log.AND}=32t_{\log.AND},$$

where m is the number of inputs, n is the bit depth of the inputs, $t_{\text{лог.I}}$ is the response time of the logical element AND, $7\log_2 n\times\ t_{\log.\text{AND}}$ is the addition time on the parallel n bit adder, $t_{Add_{7-3}}$ – addition time on 7-input single-bit adder, $t_{Add_{3-2}}$ – addition time on 3-input single-bit adder, t_{Add_8} – addition time on 8-bit parallel adder. A comparison of the addition time of an eight-input eight-bit cascade parallel-parallel adder with the addition time of an eight-input eight-bit adder using vertical and multi-operand approaches and 3- and 7-input single-bit adders shows that the addition time per adder using vertical and of multi-operand approaches and 3- and 7-input one-bit adders is reduced by approximately two times.

Reducing the time of obtaining the sum in the multi-input adder MSm is achieved through the integrated use of vertical and multi-operand approaches and multi-input single-digit adders combined according to the principle of the Wallace tree.

DISCUSSION OF RESULTS

It is proposed to implement the CNN with the specified technical parameters on the basis of a problem-oriented approach that uses the processor core with supplemented hardware accelerators that implement basic operations of the CNN.

High efficiency of hardware use for realization of two-dimensional convolution in a sliding window in real time is achieved by coordinating the duration of the conveyor cycle T_{SP_C} with the duration of the cycle of receipt of columns of input data and weights t_c, which leads to minimization of hardware costs for their implementation and, accordingly, to increase the efficiency of equipment use. The coordination of t_c with T_{SP_C} may require both an increase and a decrease in the duration of the conveyor cycle. The main ways to reduce T_{SP_C} are: the use of algorithms that reduce the number of partial products (an algorithm with the formation of partial products for the sum of two pairs of products with the analysis of one bit of factors); conveyorization of a multi-input adder by splitting it into steps; parallel inclusion of two or more devices for calculating the scalar product, the number of which is determined mainly by the time t_c of receipt of input data and weighting coefficients. The main ways to increase T_{SP_C} are: the use of algorithms that reduce the number of steps of the pipeline (Booth's algorithm, an algorithm with the formation of group partial products); implementation of two or more iterations by one step of the pipeline Algorithm.

Further directions of research are the development of a method and algorithms for end-to-end design of convolutional neural networks for recognition of biometric images. The implementation of recognition algorithms will be carried out on the basis of FPGA.

CONCLUSION

1. A parallel-flow structure of hardware for the implementation of two-dimensional convolution in a sliding window has been developed, which is focused on VLSI-implementation with high efficiency of equipment use and is the basis for the implementation of the hardware accelerator CNN.
2. The method of group summation has been improved, which, using multi-input single-digit adders combined according to the principle of the Wallace tree, provides a reduction in summation time.
3. The method of parallel-stream calculation of scalar product by forming group partial products for 4 and more digits of factors using modified Booth algorithm is proposed. This reduced the number of steps of the conveyor by 4 or more times.

4. It is shown that the development of a hardware accelerator for calculating a two-dimensional convolution in a sliding window with high efficiency of equipment use is reduced to providing calculations in real time at minimal hardware costs. This is achieved by matching the time of receipt of the columns of input data and weighting coefficients with the duration of the conveyor cycle of the accelerator.

5. The structure of hardware for realization of two-dimensional convolution in a sliding window, which is focused on VLSI-implementation with high efficiency of equipment use, has been developed.

6. Single-bit 7, 15 and 31 input adders were simulated based on the FPGA EP3C16F484 of the Cyclone III family of Altera company and an 8-input 7-bit adder was synthesized on their basis.

REFERENCES

1. Chen, Y-H., Krishna, T., Emer, J.S. & Sze, V. "Eyeriss, an energy-efficient reconfigurable accelerator for deep convolutional neural networks". *IEEE J Solid-State Circuits*. 2017; 52(1): 127–138. DOI: https://doi.org/0.1109/JSSC.2016.2616357.

2. Chen, Y-H., Krishna, T, Emer, J. S. & Sze, V. "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices". *IEEE J Emerg Sel Top Circuits Syst*. 2019; 9(2): 292–308. – Available from: https://arxiv.org/pdf/1807.07928.pdf. – [Accessed: 22, Dec. 2022].

3. Wu, R., Guo, X., Du, J. & Li, J. "Accelerating neural network inference on FPGA-based platforms – A survey". *Electronics*. 2021; 10: 1025. DOI: https://doi.org/10.3390/electronics10091025.

4. Duka, A. V. "Neural network based inverse kinematics solution for trajectory tracking of a robotic arm". *Procedia Technol*. 2014; 12: 20–27. DOI: https://doi.org/10.1016/j.protcy.2013.12.451.

5. Tsmots, I., Teslyuk, V., Kryvinska, N., Skorokhoda & O. & Kazymyra, I. "Development of a generalized model for parallel-streaming neural element and structures for scalar product calculation devices". *Journal of Supercomputing*. 2022; 79: 4820–4846. DOI: https://doi.org/10.1007/s11227-022-04838-0.

6. Nurvitadhi, E., Venkatesh, G., Sim, J., Marr, D., Huang, R., Ong Gee Hock J., Liew, Y.T., Srivatsan, K., Moss, D., Subhaschandra, S. et al. "Can FPGAs Beat GPUs in accelerating next-generation deep neural networks". In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey*, CA, USA, ACM: NY, USA, 2017. p. 5–14.

7. Trimberger, S. M. "Three ages of FPGAs: a retrospective on the first thirty years of FPGA technology". *Proc IEEE*. 2015; 103 (3): 318–331. DOI: https://doi.org/10.1109/JPROC.2015.2392104

8. Lotric, U. & Bulic, P. "Applicability of approximate multipliers in hardware neural networks". *Neurocomputing*. 2012; 96: 57–65. DOI: https://doi.org/10.1016/j.neucom.2011.09.039.

9. Sugiarto, I., Axenie, C. & Conradt, J. "FPGA-based hardware accelerator for an embedded factor graph with configurable optimization". *J Circuits Syst Comput*. 2019; 28 (02): 1950031. DOI: https://doi.org/10.1142/S0218126619500312.

10. Rau, B.R. & Fisher, J.A. "Instruction-level parallel processing: history, overview and perspective". *The Journal of Supercomputing*. 1993; 7 (1): 9–50. – Available from: https://course.ece.cmu.edu/~ece447/s15/lib/exe/fetch.php?media=ilp_history_overview_perspective.pdf. – [Accessed: 22, Dec. 2022].

11.Sohi, G. "Instruction issue logic for high-performance interruptible, multiple functional units, pipelined computers". *IEEE Trans Comput*. 1990; 39 (3): 349–359. DOI: https://doi.org/10.1109/12.48865.

12. Himavathi, S., Anitha, D. & Himavathi, S. "Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization". *IEEE Trans Neural Networks*. 2007; 18 (3): 880–888. DOI: https://doi.org/10.1109/TNN.2007.891626.

13. Oskouei, S. S. L., Golestani, H., Kachuee, M., Hashemi, M., Mohammadzade, H. & Ghiasi, S. "GPU-based acceleration of deep convolutional neural networks on mobile platforms". *Distrib Parallel Clust Comput*. 2015. – Available from: https://arxiv.org/pdf/1511.07376v1.pdf. – [Accessed: 22, Dec. 2022].

14. Huqqani, A. A., Schikuta, E., Ye, S. & Chen, P. "Multicore and GPU parallelization of neural networks for face recognition". *Procedia Comput Science*. 2013; 18: 349–358. DOI: https://doi.org/10.1016/j.procs.2013.05.198.

15. Tsmots, I., Teslyuk, V., Teslyuk, T. & Ihnatyev, I. "Basic components of neuronetworks with parallel vertical group data real-time processing". *In: Advances in Intelligent Systems and Computing II Advances in Intelligent Systems and Computing 689. Springer International Publishing.* 2018. p. 558–576. – DOI: https://doi.org/10.1007/978-3-319-70581-1_39.

16. Tsmots, I., Skorokhoda, O., Ignatyev, I. & Rabyk, V. "Basic vertical-parallel real time neural network components". *In: Proceedings of XIIth International Scientific and Technical Conference, CSIT*. Lviv: Ukraine. 2017. p. 344–347. DOI: https://doi.org/10.1109/STC-CSIT.2017.8098801.

17.Tsmots, I. G., Teslyuk, V. M. & Skorokhoda, O. V. “Device for calculating the scalar product”. Patent of Ukraine for an invention No. 124637, October 20, 2021, Byul. No. 42.

18. Li, X., Huang, H., Chen, T., Gao, H., Hu, X. & Xiong, X. “A hardware-efficient computing engine for FPGA-based deep convolutional neural network accelerator”. *Microelectronics Journal*. 2022; 128: 105547. DOI: https://doi.org/10.1016/j.mejo.2022.105547.

19. Khan, F. H., Pasha, M. A. & Masud, S. “Towards designing a hardware accelerator for 3D convolutional neural networks”. *Computers and Electrical Engineering*. 2023; 105: 108489. DOI: https://doi.org/10.1016/j.compeleceng.2022.108489.

20. Liu, Z., Xiao, X., Li, C., Ma, S. & Rangyu, D. “Optimizing convolutional neural networks on multicore vector accelerator”. *Parallel Computing*. 2022; 112, 102945. DOI: https://doi.org/10.1016/j.parco.2022.102945.

21. Ding, W., Huang, Z., Huang, Z., Tian, L., Wang, H. & Feng, S. “Designing efficient accelerator of depthwise separable convolutional neural network on FPGA”. *Journal of Systems Architecture*. 2019; 97: 278–286. DOI: https://doi.org/10.1016/j.sysarc.2018.12.008.

22. Firuzan, A., Modarressi, M., Reshadi, M. & Khademzadeh, A. “Reconfigurable Network-on-Chip based Convolutional Neural Network Accelerator”. *Journal of Systems Architecture*. 2022; 129, 102567. DOI: https://doi.org/10.1016/j.sysarc.2022.102567.

23.George, B., Omer, O. J., Choudhury, Z. & Subramoney, A. V. “A Unified Programmable Edge Matrix Processor for Deep Neural Networks and Matrix Algebra”. *ACM Transactions on Embedded Computing Systems.* 2022; 21(5): 63:1–63:30. DOI: https://doi.org/10.1145/3524453.

24.Majumder, K. & Bondhugula U. “A flexible FPGA accelerator for convolutional neural networks”. arXiv:1912.07284. 2019. DOI: https://doi.org/10.48550/arXiv.1912.07284.

25.Choudhury, Z., Shrivastava, S, Ramapantulu, L. & Purini, S. “An FPGA Overlay for CNN Inference with Fine-grained Flexible Parallelism”. *ACM Transactions on Architecture and Code Optimization*. 2022; 19(3), Article No. 34: 1–26. DOI: https://doi.org/10.1145/3519598.

26.Wu, R., Guo, X., Du, J. & Li, J. “Accelerating neural network inference on FPGA-based platforms – A survey”. *Electronics*. 2021; 10 (9): 1025. DOI: https://doi.org/10.3390/electronics10091025.

27.Nurvitadhi, E., Venkatesh, G., Sim, J., Marr, D., et al. “Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?”. In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2017: 5–14. – Available from: https://jaewoong.org/pubs/fpga17-next-generation-dnns.pdf. – [Accessed: 22, Dec. 2022].

28. Antoniuk, V. V., Drozd, M. O. & Drozd, O. B. “Power-oriented checkability and monitoring of the current consumption in FPGA projects of the critical applications”. A*pplied Aspects of Information Technology*. 2019; 2 (2): 105–114. DOI: https://doi.org/10.15276/aait.02.2019.2.

29. Drozd, O. V., Rucinski, A., Zashcholkin, K. V., Drozd, M. O. & Sulima, Yu.Yu. “Improving FPGA components of critical systems based on natural version redundancy”. *Applied Aspects of Information Technology*. 2021; 4 (2): 168–177. DOI: https://doi.org/10.15276/aait.02.2021.4.

30. Berezsky, O. M. & Liashchynskyi, P. B. “Comparison of generative adversarial networks architectures for biomedical images synthesis”. *Applied Aspects of Information Technology*. 2021; 4 (3): 250–260. DOI: https://doi.org/10.15276/aait.03.2021.4.

Conflicts of Interest: the authors declare no conflict of interest

Received 25.01.2023
Received after revision 28.03.2023
Accepted 04.04.2023

DOI: https://doi.org/10.15276/aait.06.2023.1
УДК 004.032.26

Методи та апаратні засоби для прискорення роботи згорткової нейронної мережі

Цмоць Іван Григорович[1)]
ORCID: https://orcid.org/0000-0002-4033-8618; ivan.tsmots@gmail.com. Scopus Author ID: 24484154400
Березький Олег Миколайович[2)]
ORCID: https://orcid.org/0000-0001-9931-4154; olber62@gmail.com. Scopus Author ID: 6505609877

Березький Микола Олегович[2)]
ORCID: https://orcid.org/0000-0001-6507-9117; mykolaberezkyy@gmail.com. Scopus Author ID: 58020232600
[1)] Національний університет «Львівська політехніка», вул. С. Бандери, 12. Львів, 79000, Україна
[2)] Західноукраїнський національний університет, вул. Львівська, 11. Тернопіль, 46009, Україна

АНОТАЦІЯ

Проаналізовано і виділено три основні підходи до побудови комп'ютерних систем: програмний, апаратний та проблемно-орієнтований. Вибрано для реалізації ЗНМ проблемно-орієнтований підхід. Цей підхід використовує процесорне ядро з апаратними прискорювачами, що реалізують базові операції ЗНМ. Розробку комп'ютерних систем для реалізації ЗНМ доцільно здійснювати на основі інтегрованого підходу. Цей підхід включає сучасну елементу базу, існуючі апаратні та програмні засоби для реалізації ЗНМ; методи і алгоритми реалізації ЗНМ; методи, алгоритми і НВІС-структури для реалізації базових операцій ЗНМ; методи та засоби автоматизованого проектування апаратних і програмних засобів орієнтованих на реалізацію ЗНМ комп'ютерних систем. Для розробки комп'ютерних систем для реалізації ЗНМ вибрано підхід, який включає: змінний склад обладнання; використання базису елементарних арифметичних операцій; організація процесу обчислення скалярного добутку як виконання єдиної операції; конвеєризації і просторового паралелізму; локалізації і спрощення зв'язків між сходинками конвеєра; узгодження часу формування вхідних даних і вагових коефіцієнтів з тривалістю конвеєрного такту. Показано, що для зменшення часу опрацювання зображень великого обсягу найдоцільніше використати паралельно-потокову НВІС-реалізацію базових операцій. Вибрано модифікованиий алгоритм Бута для формування часткових добутків у паралельно-потоковому пристрою обчислення скалярного добутку, що забезпечило зменшення у два рази кількості сходинок конвеєра. Вдосконалено метод групового підсумовування, який за рахунок використання багатовходових однорозрядних суматорів, об'єднаних за принципом дерева Уоллеса, забезпечує зменшення часу підсумовування. Розроблено метод паралельно-потокового обчислення скалярного добутку у ковзаючому вікні, який за рахунок узгодження часу надходження стовпчиків вхідних даних і вагових коефіцієнтів з тривалістю конвеєрного такту забезпечує високу ефективність використання обладнання та обчислення у реальному часі. Визначено основні шляхи узгодження часу надходження стовпчиків вхідних даних і вагових коефіцієнтів з тривалістю конвеєрного такту роботи апаратних засобів, які реалізують двовимірну згортку. Розроблено структуру апаратних засобів для реалізації двовимірної згортки у ковзаючому вікні, яка орієнтована на НВІС-реалізацію з високою ефективністю використання обладнання. Вибрано для реалізації апаратних прискорювачів програмовані логічні інтегральні схеми. Розроблено та відмодельовано на базі FPGA EP3C16F484 сімейства Cyclone III фірми Altera однорозрядні 7, 15 і 31 входові суматори та синтезовано на їх базі 8-входовий 7-розрядний суматор.

Ключові слова: Згорткові нейронні мережі; апаратний прискорювач; проблемно-орієнтований підхід; паралельно-потокова реалізація; багатовходовий суматор; скалярний добуток; двовимірна згортка

ABOUT THE AUTHORS

Ivan G. Tsmots - Doctor of Engineering Sciences, Professor. Department of Automated Control Systems,
Lviv Polytechnic National University, 12, Bandera Str. Lviv, 79000, Ukraine
ORCID: https://orcid.org/0000-0002-4033-8618; ivan.tsmots@gmail.com. Scopus Author ID: 24484154400
Research field: Intelligent information technologies; multi-level control systems and specialized tools for parallel processing of signals and images

Цмоць Іван Григорович - доктор технічних наук, професор кафедри Автоматизованих систем управління. Національний університет «Львівська політехніка», вул. С. Бандери, 12. Львів, 79000, Україна

Oleh M. Berezsky - Doctor of Engineering Sciences, Professor. Department of Computer Engineering.
West Ukrainian National University, 11, Lvivska Str. Ternopil, 46009, Ukraine
ORCID: https://orcid.org/0000-0001-9931-4154; olber62@gmail.com. Scopus Author ID: 6505609877
***Research field*:** Artificial intelligence; computer vision; machine learning; artificial neural networks

Березький Олег Миколайович – доктор технічних наук, професор кафедри Комп'ютерної інженерії. Західноукраїнський національний університет, вул. Львівська, 11. Тернопіль, 46009, Україна

Mykola O. Berezkyy - graduate student of the Department of Computer Engineering.
West Ukrainian National University, 11, Lvivska Str. Ternopil, 46009, Ukraine
ORCID: https://orcid.org/0000-0001-6507-9117; mykolaberezkyy@gmail.com. Scopus Author ID: 58020232600
Research field: Computer vision; machine learning

Березький Микола Олегович - аспірант кафедри Комп'ютерної інженерії.
Західноукраїнський національний університет, вул. Львівська, 11. Тернопіль, 46009, Україна