Aleksandr CARIOW, Galina CARIOWA
WEST POMIERANIAN UNIVERSITY OF TECHNOLOGY, SZCZECIN,
Żołnierska St. 49, 71-210 Szczecin

A parallel hardware-oriented algorithm for constant matrix-vector multiplication with reduced multiplicative complexity

Prof. dr hab. inż. Aleksandr CARIOW

He received the Candidate of Sciences (PhD) and Doctor of Sciences degrees (DSc or Habilitation) in Computer Sciences from LITMO of St. Petersburg, Russia in 1984 and 2001, respectively. In September 1999, he joined the faculty of Computer Sciences at the West Pomeranian University of Technology, where he is currently a professor and chair of the Department of Computer Architectures and Telecommunications. His research interests include digital signal processing algorithms, VLSI architectures and data processing parallelization.

e-mail: atariov@wi.zut.edu.pl

Dr Galina CARIOWA

She received the MSc degrees in mathematics from Moldavian State University, Chişinău in 1976 and PhD degree in computer science from West Pomeranian University of Technology in 2007. She is currently working as an assistant professor of the Department of Multimedia Systems. She is also an Associate-Editor of World Research Journal of Transactions on Algorithms. Her scientific interests include numerical linear algebra and digital signal processing algorithms, VLSI architectures, and data processing parallelization.

e-mail: gtariova@wi.zut.edu.pl

Abstract

This paper presents the algorithmic aspects of organization of a low-complexity fully parallel processor unit for constant matrix-vector products computing. To reduce the hardware complexity (number of two-operand multipliers), we exploit the Winograd's inner product calculation approach. We show that by using this approach, the computational process of calculating the constant matrix-vector product can be structured so that it eventually requires fewer multipliers than the direct implementation of matrix-vector multiplication.

Keywords: constant coefficient matrix-vector multiplier, hardware complexity reduction, FPGA implementation.

Równoległy sprzętowo zorientowany algorytm mnożenia macierzy stałych przez wektor ze zredukowaną złożonością multiplikatywną

Streszczenie

W pracy został przedstawiony sprzętowo-zorientowany algorytm wyznaczania iloczynu wektora przez macierz stałych. W odróżnieniu od implementacji naiwnego sposobu zrównoleglenia obliczeń wymagającego N^2 układów mnożących proponowana równoległa struktura wymaga tylko $N(M+1)/2$ takich układów. A ponieważ układ mnożący pochłania znacznie więcej zasobów sprzętowych platformy implementacyjnej niż sumator, to minimalizacja liczby tych układów podczas projektowania dedykowanych układów obliczeniowych jest sprawą nadrzędną. Idea syntezy algorytmu oparta jest na wykorzystaniu do wyznaczania cząstkowych iloczynów skalarnych metody S. Winograda. Zaprezentowany w artykule algorytm może być z powodzeniem zastosowany do akceleracji obliczeń w podsystemach cyfrowego przetwarzania danych zrealizowanych na platformach FPGA oraz zaimplementowany w dowolnym środowisku sprzętowym, na przykład zrealizowana w postaci układu ASIC. W tym ostatnim przypadku niewątpliwym atutem wyróżniającym przedstawione rozwiązanie jest to, że zaprojektowany w ten sposób układ będzie zużywać mniej energii oraz wydzielać mniej ciepła.

Słowa kluczowe: układ mnożenia macierzy, redukcja złożoności sprzętowej, implementacja na FPGA.

1. Introduction

This paper describes the structure and principles of a processor unit to calculate the constant matrix-vector product (CMVP) with the reduced number of multipliers. In the general case a fully parallel hardware implementation of matrix-vector multiplication requires N^2 multipliers. In the case where the matrix elements are constants, we can use encoders instead of multipliers. This solution greatly simplifies implementation, reduces the power dissipation and lowers the price of the device. On the other hand, when we deal with FPGA chips that contain several tens or even hundreds of embedded multipliers, the building and using of additional encoders instead of multipliers is irrational. In this case, it would be unreasonable to refuse the possibility of using embedded multipliers. Nevertheless, the number of on-chip multipliers is always limited, and this number may sometimes not be enough to implement a high-speed fully parallel vector-matrix multiplier. Therefore, finding ways to reduce the number of multipliers in the implementation of vector-matrix multiplier is an extremely urgent task. It is from this perspective, we consider the below solution.

It is known that a main kernel during the calculation of vector-matrix product is the inner product operation. In calculating a vector-matrix product the inner product of vectors is computed as many times as rows the matrix contains. To reduce the number of two-input multipliers required to implement such a matrix-vector multiplier, we use the Winograd's method for calculating inner product of two vectors. The order of the multiplicative complexity (number of two-input multipliers) is commonly used to measure and compare the efficiency of the algorithms since multipliers are intrinsically more complicated than the other arithmetical units.

2. Statement of the problem

Let $\mathbf{X}_{N\times 1}=[x_0,x_1,...,x_{N-1}]^{\mathrm{T}}$ and $\mathbf{Y}_{M\times 1}=[y_0,y_1,...,y_{M-1}]^{\mathrm{T}}$ - are N-point and M-point one dimensional data vectors respectively, and $m=0,1,...,M-1$, $n=0,1,...,N-1$.

The problem is to calculate a product

$$\mathbf{Y}_{M\times 1}=\mathbf{A}_{M\times N}\mathbf{X}_{N\times 1}, \tag{1}$$

where

$$\mathbf{A}_{M\times N}=\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix},$$

and $a_{m,n}$ - are l-bit fixed-point constants.

Direct and full parallel implementation of matrix product (1) requires NM multipliers and $M(N-1)$ adders. Some effective realizations of this operation for the constant coefficient matrix have been reported in the literature [5-9], but the analysis of these approaches shows that possibilities of developing effective CMVP algorithms are not exhausted. By exploiting some rationalization solutions based on using the Winograd's inner product algorithm the number of necessary multipliers for fully-parallel implementation could be decreases. In this paper, we describe in detail and show how it can be done.

3. Synthesis of the algorithm

According to the Winograd's formula for inner product calculation each element of vector $\mathbf{Y}_{M\times 1}$ can be calculated as follows [10]:

$$y_m = \sum_{k=0}^{\frac{N}{2}-1}[(a_{m,2k}+x_{2k+1})(a_{m,2k+1}+x_{2k})]-c_m-\xi(N) \qquad (2)$$

where

$$c_m = \sum_{k=0}^{\frac{N}{2}-1} a_{m,2k}\cdot a_{m,2k+1} \text{ and } \xi(N) = \sum_{k=0}^{\frac{N}{2}-1} x_{2k}\cdot x_{2k+1}$$

if N is even.

Here it should be emphasized that if c_m can be calculated in advance, the calculation of $\xi(N)$ requires the implementation of $N/2$ multiplications.

Equation (2) completely determines the list of operations that must be performed in accordance with the Winograd's inner product method, but no sequence of computations or formats of the data being processed in this formula is defined. Also the provided formula does not take into account the possibility of parallel computing, which is available. Therefore, to describe the proposed algorithm, we use the matrix notation, which is more suitable for describing the parallel computations. Below we consider the synthesis of our algorithm in detail.

First, we split vector $\mathbf{X}_{N\times 1}$ into two vectors $\mathbf{X}^{(1)}_{\frac{N}{2}\times 1}$ and $\mathbf{X}^{(2)}_{\frac{N}{2}\times 1}$ containing only even-numbered and only odd-numbered elements, respectively:

$$\mathbf{X}^{(1)}_{\frac{N}{2}\times 1} = [x_0, x_2, ..., x_{N-2}]^{\mathrm{T}},\ \mathbf{X}^{(2)}_{\frac{N}{2}\times 1} = [x_1, x_3, ..., x_{N-1}]^{\mathrm{T}}.$$

Then from the elements of the matrix $\mathbf{A}_{M\times N}$ we form two super-vectors of data:

$$\mathbf{A}^{(1)}_{\frac{MN}{2}\times 1} = [\widehat{\mathbf{A}}^{(0)}_{M\times 1}, \widehat{\mathbf{A}}^{(1)}_{M\times 1}, ..., \widehat{\mathbf{A}}^{(\frac{N}{2}-1)}_{M\times 1}]^{\mathrm{T}},$$

$$\mathbf{A}^{(2)}_{\frac{MN}{2}\times 1} = [\breve{\mathbf{A}}^{(0)}_{M\times 1}, \breve{\mathbf{A}}^{(1)}_{M\times 1}, ..., \breve{\mathbf{A}}^{(\frac{N}{2}-1)}_{M\times 1}]^{\mathrm{T}},$$

where

$$\widehat{\mathbf{A}}^{(k)}_{M\times 1} = [a_{0,2k+1}, a_{1,2k+1}, ..., a_{M-1,2k+1}]^{\mathrm{T}},$$

$$\breve{\mathbf{A}}^{(k)}_{M\times 1} = [a_{0,2k}, a_{1,2k}, ..., a_{M-1,2k}]^{\mathrm{T}}.$$

We also define the vectors

$$\mathbf{C}_{M\times 1} = [c_0, c_1, ..., c_{M-1}]^{\mathrm{T}}, \text{ and } \mathbf{E}_{M\times 1} = [\underbrace{\xi(N), \xi(N), ..., \xi(N)}_{M \text{ times}}].$$

Next, we introduce some auxiliary matrices:

$$\mathbf{P}_{\frac{MN}{2}\times\frac{N}{2}} = (\mathbf{I}_{\frac{N}{2}} \otimes \mathbf{1}_{M\times 1}),\ \mathbf{\Sigma}_{M\times\frac{MN}{2}} = (\mathbf{1}_{1\times\frac{N}{2}} \otimes \mathbf{I}_M).$$

where $\mathbf{1}_{M\times N}$ - is an $M\times N$ matrix of ones (a matrix where every element is equal to one), $\mathbf{I}_N$ - is an identity $N\times N$ matrix and signs „$\otimes$" denote the tensor product of two matrices [11].

Using the above matrices the rationalized computational procedure for calculating the constant matrix-vector product can be written as follows:

$$\mathbf{Y}_{M\times 1} = \mathbf{E}_{M\times 1} + \{\mathbf{C}_{M\times 1} + [\mathbf{\Sigma}_{M\times\frac{MN}{2}} \mathbf{D}_{\frac{MN}{2}} (\mathbf{A}^{(1)}_{\frac{MN}{2}\times 1} + \mathbf{P}_{\frac{MN}{2}\times\frac{N}{2}} \mathbf{X}^{(1)}_{\frac{N}{2}\times 1})]\}$$

where

$$\mathbf{D}_{\frac{MN}{2}} = diag(\mathbf{D}^{(0)}_M, \mathbf{D}^{(1)}_M, ..., \mathbf{D}^{(\frac{N}{2}-1)}_M)$$

and

$$\mathbf{D}^{(k)}_M = diag(s^{(k)}_0, s^{(k)}_1, ..., s^{(k)}_{M-1}).$$

If the elements of $\mathbf{D}_{\frac{MN}{2}}$ are placed vertically without disturbing the order and written in the form of the vector $\mathbf{S}_{\frac{MN}{2}\times 1}$, then they can be calculated using the following vector-matrix procedure:

$$\mathbf{S}_{\frac{MN}{2}\times 1} = \mathbf{A}^{(2)}_{\frac{MN}{2}\times 1} + \mathbf{P}_{\frac{MN}{2}\times\frac{M}{2}} \mathbf{X}^{(2)}_{\frac{N}{2}\times 1}.$$

Fig. 1 shows the data flow diagram representation of the rationalized algorithm for computation of the constant matrix-vector product for the case $M = N = 8$.

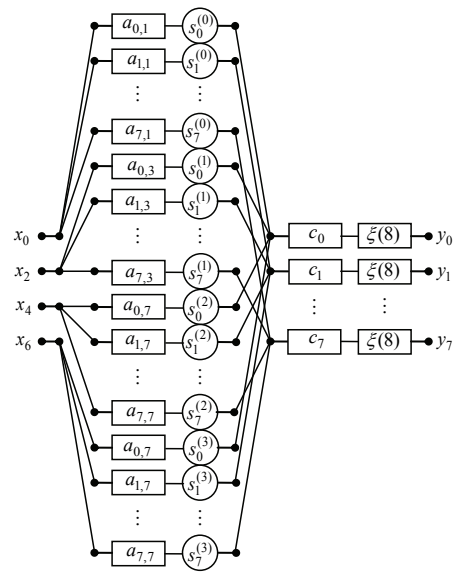

Fig. 1. Data flow diagram for the rationalized constant matrix-vector multiplication algorithm for N=M=8

Rys. 1. Model grafostrukturalny, reprezentujący proponowany algorytm mnożenia macierzy stałych przez wektor zmiennych dla N=M=8

Fig. 2 shows the data flow diagram of the process for calculating the matrix $\mathbf{D}_{\frac{MN}{2}}$ (vector $\mathbf{S}_{\frac{MN}{2}\times 1}$) elements for this example. The straight lines in the figures denote the operations of data transfer. The circles in these figures show the operation of multiplication by the number inscribed inside the circle. In turn, the rectangles indicate the addition with the number (variable or constant) inscribed inside the rectangle. In this paper, the data

flow diagrams are oriented from left to right. The points where lines converge denote summation. We use the usual lines without arrows on purpose, so as not to clutter the picture.

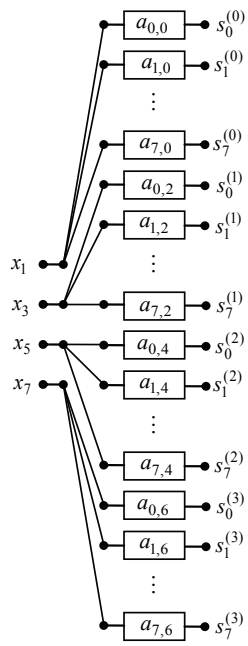

Fig. 2. Data flow diagram for calculating the elements of diagonal matrix $\mathbf{D}_{MN/2}$ for N=M=8

Rys. 2. Model grafostrukturalny, opisujący algorytm wyznaczania elementów macierzy $\mathbf{D}_{MN/2}$ dla N=M=8

4. Discussion of hardware implementation complexity

We calculate how many multipliers and adders are required, and compare this with the number required for a fully parallel direct implementation of the matrix–vector product in Eq. (1). The number of conventional two-input multipliers required using the proposed algorithm is $N(M+1)/2$. Thus using the proposed algorithm the number of multipliers to implement the constant matrix-vector product is dramatically reduced. Additionally our algorithm requires $M(N+1)$ one-input adders with constant numbers (ordinary encoders), M two-input adders, and $M+1$ $(N/2)$ -input adders.

The number of conventional two-input multipliers required when using the fully parallel implementation of the "schoolbook" method for matrix-vector multiplication is MN. This implementation also requires the M N -inputs adders. Thus, our proposed algorithm saves almost 50% two-input embedded multipliers (through the application of the Winograd's formula for calculation of the scalar product of vectors) but it significantly increases the number of adders compared with the direct method fully-parallel implementation. For applications where the "cost" of a multiplication is greater than that of an addition, the new algorithm is always more computationally efficient than the direct evaluation of the matrix-vector product. This allows concluding that the suggested solution may be useful in a number of cases and have practical application allowing minimizing the costs of constant matrix-vector multiplier's hardware implementation.

5. Concluding remarks

The paper presents a new hardware-oriented algorithm for computing the constant matrix-vector multiplication. To reduce the hardware complexity (number of two-operand multipliers), we exploit the Winograd's inner product calculation approach. This allows the effective use of parallelization of vector-matrix multiplication on the one hand and results in a reduction in hardware implementation cost on the other hand. If the FPGA-chip already contains embedded multipliers, their number is always limited. This means that if the implemented algorithm contains a large number of multiplications, the developed processor may not always fit into the chip. So, the implementation of the proposed in this paper algorithm on the base of FPGA circuits that have built-in binary multipliers, also allows saving the number of blocks or realizing the whole matrix-vector multiplying unit with the use of a smaller number of simpler and cheaper FGPA circuits. For instance, a completely parallel implementation of an 8×8 constant coefficient matrix-vector multiplier using the schoolbook (naïve) method requires six FGPA-chips Spartan-3 XC3S250E, while the implementation of the proposed algorithm requires only three of them or only one chip XC3S1600E.

6. References

[1] Amira A. Bouridane and Milligan P.: Accelerating matrix product on reconfigurable hardware for signal processing, in Proc. 11th Int. Conf. Field-Programmable Logic Appl. (FPL), 2001, pp. 101–111.

[2] Oscar Gustafsson, Henrik Ohlsson, and Lars Wanhammar: Low-Complexity Constant Coefficient Matrix Multiplication Using a Minimum Spanning Tree Approach, Proceedings of the 6th Nordic Signal Processing Symposium - NORSIG 2004, June 9 - 11, 2004, Espoo, Finland, pp. 141-144.

[3] Jang J., Choi S. and Prasanna V. K.: Energy-efficient matrix multiplication on FPGAs, in Proc. Int. Conf. Field Programmable Logic Appl., 2002, pp. 534–544.

[4] Jang J. W., Choi S. and Prasanna V. K.: Area and time efficient implementations of matrix multiplication on FPGAs, in Proc. IEEE Int. Conf. Field Programmable Technol., 2002, pp. 93–100.

[5] Syed M. Qasim, Ahmed A. Telba and Abdulhameed Y. AlMazroo: FPGA Design and Implementation of Matrix Multiplier Architectures for Image and Signal Processing Applications, IJCSNS International Journal of Computer Science and Network Security, 2010, vol. 10 No.2, pp. 168-176.

[6] Nicolas Boullis and Arnaud Tisserand: Some Optimizations of Hardware Multiplication by Constant Matrices, IEEE Transactions on Computers, 2005, vol. 54, no. 10, pp. 1271-1282.

[7] Kinane Andrew, Muresan Valentin, O'Connor and Noel E.: Optimization of constant matrix multiplication operation hardware using a genetic algorithm. In: EvoHOT 2006 - 3rd European Workshop on Evolutionary Computation in Hardware Optimization, 10-12 April 2006, Budapest, Hungary. pp. 296-307.

[8] Boullis N. and Tisserand A.: Some Optimizations of Hardware Multiplication by Constant Matrices, IEEE Trans. Comput., 2005, vol. 54, no. 10, pp. 1271-1282.

[9] Andrew Kinane, Valentin Muresan and Noel O'Connor: Towards an Optimised VLSI Design Algorithm for the Constant Matrix Multiplication Problem, ISCAS 2006, pp. 5111-5114.

[10] Winograd S.: A New Algorithm for Inner Product, IEEE Transactions on Computers - TC , 1968, vol. C-17, no. 7, pp. 693-694,

[11] Willi-Hans Steeb, Yorick Hardy: Matrix Calculus and Kronecker Product: A Practical Approach to Linear and Multilinear Algebra, World Scientific Publishing Company; 2 edition (March 24, 2011), 324 pages.

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*