BULETINUL ŞTIINŢIFIC

al

Universităţii „POLITEHNICA" din Timişoara, România

Seria ELECTRONICĂ ŞI TELECOMUNICAŢII

Număr special dedicat Simpozionului
de "Electronică şi Telecomunicaţii, ETc 2004"

Timişoara, 22-23 octombrie, 2004

SCIENTIFIC BULLETIN

of

the „POLITEHNICA" University of Timişoara, Romania

Transactions on ELECTRONICS AND COMMUNICATIONS

Tomul 49(63), Fascicola 1, 2004
ISSN 1583-3380

EDITURA POLITEHNICA

Electronic Circuits

Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

An Efficient Linear Systolic Array Architecture for a Memory-Based VLSI Implementation of Type III Generalized Hartley Transform

Doru Florin Chiper[1] and C. Comşa[1]

Abstract- An efficient design approach for a VLSI array of a prime length type III Generalized Discrete Hartley Transform (GDHT) is proposed. The presented approach uses an appropriate decomposition of type III GDHT into two cyclic convolutions having the same length and structure. The two computational structures can be implemented in parallel using the same hardware structure based on using bi-port ROMS. Using an appropriate hardware sharing technique and a VLSI architecture based on small bi-port ROMs we can obtain high computing speed with low hardware complexity and low I/O costs together with all the other advantages of the systolic array implementation of cyclic convolution as regular and modular structures with local connections.

Keywords: Generalized discrete Hartley transform, systolic algorithms, Systolic architectures, Memory-based architectures

I. INTRODUCTION

The generalized discrete Hartley transform (GDHT) is used to efficiently replace the generalized discrete Fourier transform (GDFT) when the input sequence is real. It has the same useful applications as GDFT in many digital signal processing applications as filter banks, computing convolutions and the fast computation of the discrete Fourier transform or in signal representation.

There are many software implementations of the type III GDHT but no efficient hardware implementation has been proposed until now. Due to the fact that GDHT is computational intensive it is necessary to develop efficient VLSI algorithms to meet the requirements of a real-time application. The efficiency of a VLSI algorithm is based more on the communication complexity than on the computational one. Thus, as already been observed, fast algorithms although characterized by a small number of multiplications are not suitable for a VLSI implementation. The rational of this is the fact that the data flow plays a central role in designing a VLSI architecture. Thus, the use of regular and modular computational structures as cyclic convolution and circular correlation has been proved to offer good implementation solutions for the discrete transforms using systolic arrays leading to low I/O costs and reduced hardware complexity, high speed and a regular and modular hardware structure.

The advantages of a systolic array implementation are more evident when a large number of processing elements can be integrated on the same chip. The number of processing elements on a chip can be increased when is possible to reduce the hardware complexity of the processing elements. One way to achieve this is to efficiently replace the chip area consuming multipliers with small ROMs which offer also the advantages of a high modularity and regularity. In order to efficiently replace multipliers by ROMs we have to appropriate reformulate the VLSI algorithms such as in each multiplication operation one operand to be fixed.

In this paper, an efficient VLSI architecture for a linear systolic array implementation of a prime-length type III Generalized Hartley Transform is proposed. It employs a systolic array algorithm that can be efficiently implemented on a linear systolic array using small bi-port ROMs to replace multipliers. The proposed systolic algorithm uses an appropriate decomposition of the type III GDHT into two cyclic convolution structures, with the same length and structure that can be computed in parallel. The proposed decomposition approach is based on using auxiliary input and output sequences and the appropriate reordering of these sequences using the properties of the Galois Field of the indexes. Thus, it is possible to use the advantages of the systolic array implementations of the cyclic convolution as high speed, low I/O cost and reduced hardware complexity with a high regularity and modularity to obtain an efficient VLSI architecture.

[1] Faculty of Electronics and Telecommunications, Bd. Carol I Nr. 11, 6600, Iasi.

Using the data-dependence graph-based procedure we can obtain two linear systolic arrays that can be efficiently unified using an appropriate hardware-sharing technique. The preprocessing stage is used to convert the input sequence using some multiplication and data reordering operations into two appropriate auxiliary ones that can be processed using a cyclic convolution structure. The post-processing stage is used to convert two auxiliary output sequences into the final output sequence using some recursive computations and data reordering operations. The computation complexity of the operations implemented in the pre- and post-processing stages is of O(N) as opposed to that of the hardware kernel that implements the cyclic convolution operations which is $O(N^2)$. The tag-control scheme is used to control the loading and draining of the data sequences into the internal registers of the systolic array in such a manner that we can load and drain the input and output data using only I/O channels placed at the two ends of the linear array. Due to the fact that in each multiplication one operand has been made fixed, we can efficiently replace the multipliers of the obtained VLSI array architecture with small ROMs. Moreover, due to the fact that the two multipliers in each processing element have a fixed operand that is the same for both multipliers, we can use bi-port ROMs to implement the multiplications of the two cyclic convolution structures.

Thus, using an appropriate hardware sharing technique and choosing a linear systolic array as a VLSI architecture paradigm, we can obtain high computing speed with a low I/O cost and hardware complexity, together with all the other advantages of the linear systolic array implementations of the cyclic convolution structures as regularity, modularity and local connectivity with I/O channels placed only at the two ends of the array. Moreover, replacing the multipliers with ROMs will further increase the degree of regularity and modularity of the proposed VLSI architecture.

II. A VLSI ALGORITHM FOR TYPE III GDHT

The type III GDHT for the input sequence $\{x(i): i = 0,..., N-1\}$ is defined as:

$$Y(k) = \sum_{i=0}^{N-1} x(i) \cdot cas[(2k+1)i\alpha] \quad (1)$$

where:

$$cas(\theta) = \cos(\theta) + \sin(\theta) \quad (2)$$

$$\alpha = \pi / 2N \quad (3)$$

We can compute the output sequence as follows:

$$Y(k) = x(0) + H_C(k) + H_S(k) \quad (4)$$

$$\text{for } k = 0,..., N-1$$

$$H_C(0) = \sum_{i=1}^{(N-1)/2} [x_C(\varphi(i)) + x_C(\varphi(i + (N-1)/2))] \quad (5)$$

$$H_S(0) = \sum_{i=1}^{(N-1)/2} [x_S(\varphi(i)) + x_S(\varphi(i + (N-1)/2))] \quad (6)$$

with:

$$x_C(i) = x(i) \cdot \cos(i \cdot 2\alpha) \quad (7)$$

$$x_S(i) = x(i) \cdot \sin(i \cdot 2\alpha) \quad (8)$$

$$H_C(k) = 2T_C(k) - H_C(k-1) \quad (9)$$

$$H_S(k) = 2T_S(k) + H_S(k-1) \quad (10)$$

with:

$$T_C(\varphi(k)) = \sum_{i=1}^{(N-1)/2} [x_C(\varphi(i)) + x_C(\varphi(i + (N-1)/2))] \times \cos[\varphi(i+k) \cdot 4\alpha] \quad (11)$$

$$T_S(\varphi(k)) = \sum_{i=1}^{(N-1)/2} [x_S(\varphi(i)) + x_S(\varphi(i + (N-1)/2))] \times \cos[\varphi(i+k) \cdot 4\alpha] \quad (12)$$

where:

$$\varphi(k) = < g^k >_N \quad (13)$$

with $< \bullet >$ the result of the modulo operation.

III. AN EXAMPLE

In order to illustrate our approach, we will consider an example of a 1-D type III generalized discrete Hartley transform with the length N=7 and the primitive root g=3. In this case, the equations (11) and (12) can be computed in a matrix-vector product form as shown in (14) and (15).

The auxiliary output sequence $\{T_C(k): k = 1,2,..., N-1\}$ can be computed using a cyclic convolution form as shown in (14):

$$\begin{bmatrix} T_C(3) \\ T_C(2) \\ T_C(6) \\ T_C(4) \\ T_C(5) \\ T_C(1) \end{bmatrix} =$$

$$= \begin{bmatrix} x_C(1)+x_C(6) & x_C(3)+x_C(4) & x_C(2)+x_C(5) \\ x_C(5)+x_C(2) & x_C(1)+x_C(6) & x_C(3)+x_C(4) \\ x_C(4)+x_C(3) & x_C(5)+x_C(2) & x_C(1)+x_C(6) \\ x_C(6)+x_C(1) & x_C(4)+x_C(3) & x_C(5)+x_C(2) \\ x_C(2)+x_C(5) & x_C(6)+x_C(1) & x_C(4)+x_C(3) \\ x_C(3)+x_C(4) & x_C(2)+x_C(5) & x_C(6)+x_C(1) \end{bmatrix} \times$$

$$\times \begin{bmatrix} \cos(12\alpha) \\ \cos(8\alpha) \\ \cos(4\alpha) \end{bmatrix} \qquad (14)$$

The auxiliary output sequence $\{T_S(k): k = 1,2,...,N-1\}$ can be computed using a cyclic convolution form as shown in (15):

$$\begin{bmatrix} T_S(3) \\ T_S(2) \\ T_S(6) \\ T_S(4) \\ T_S(5) \\ T_S(1) \end{bmatrix} =$$

$$= \begin{bmatrix} x_S(1)+x_S(6) & x_S(3)+x_S(4) & x_S(2)+x_S(5) \\ x_S(5)+x_S(2) & x_S(1)+x_S(6) & x_S(3)+x_S(4) \\ x_S(4)+x_S(3) & x_S(5)+x_S(2) & x_S(1)+x_S(6) \\ x_S(6)+x_S(1) & x_S(4)+x_S(3) & x_S(5)+x_S(2) \\ x_S(2)+x_S(5) & x_S(6)+x_S(1) & x_S(4)+x_S(3) \\ x_S(3)+x_S(4) & x_S(2)+x_S(5) & x_S(6)+x_S(1) \end{bmatrix} \times$$

$$\times \begin{bmatrix} \cos(12\alpha) \\ \cos(8\alpha) \\ \cos(4\alpha) \end{bmatrix} \qquad (15)$$

where:

$$x_C(i) = x(i) \cdot \cos(i \cdot 2\alpha) \qquad (16)$$

$$x_S(i) = x(i) \cdot \sin(i \cdot 2\alpha) \qquad (17)$$

We then recursively compute the auxiliary output sequences $\{H_C(k): k = 1,2,...,N-1\}$ and $\{H_C(k): k = 1,2,...,N-1\}$ using equations (5),(6) and (9),(10). Finally, we compute the output sequence as following:

$$\begin{bmatrix} Y_{III}(0) \\ Y_{III}(1) \\ Y_{III}(2) \\ Y_{III}(3) \\ Y_{III}(4) \\ Y_{III}(5) \\ Y_{III}(6) \end{bmatrix} = \begin{bmatrix} x(0)+H_C(0)+H_S(0) \\ x(0)+H_C(1)+H_S(1) \\ x(0)+H_C(2)+H_S(2) \\ x(0)+H_C(3)+H_S(3) \\ x(0)+H_C(4)+H_S(4) \\ x(0)+H_C(5)+H_S(5) \\ x(0)+H_C(6)+H_S(6) \end{bmatrix} \qquad (18)$$

IV. THE VLSI ARCHITECTURE FOR 1-D TYPE III GDHT

For sake of simplicity of our discussion we will restrict our discussion to the particular case N=7 and g=3.

Based on the data dependence graph method presented in [9] and the tag control scheme [10] we can obtain two linear systolic arrays with the same form and length. Using an appropriate hardware-sharing technique we can obtain the systolic array from fig.1.

The elements located in the same diagonal line in the matrices from equations (14) and (15) are the same. This property leads to minimize the number of I/O channels and their bandwidth, due to the specific structure of the cyclic convolution where the elements located on a given diagonal line in the matrices of (14) and (15) are the same. This reduction in I/O cost is achieved since each element introduced into the systolic array is used in all the processing elements of the hardware core. Also, all the input and the output channels are placed at one of the two ends of the array and their number is independent of the transform length N.

As can be seen form equations (14) and (15) one operand in each multiplier can be fixed. Moreover, as can be seen from fig.1 and 2 this fixed operand is the same for the two multipliers in each processing elements. This feature allows us to introduce a dual-port ROM-based implementation technique to replace the two multipliers in each processing element. Since the ROM tables necessary to implement the two multipliers in each processing element are the same, we can use only one dual-port ROM to implement both multipliers in each processing element with a memory of 2^L, where L is the number of bits necessary to represent one operand, instead of two

such ROMs. This results in substantial reduction in the hardware cost. Using the partial sum technique [11] we can further reduce the ROM size to $2^{(L/2+1)}$ memory words at a cost of two adders. Due to the fact that the hardware structure used to implement the type III GDHT transform is a synchronous one, the control structure used to avoid the conflicts in the accessing the content of the shared memory can be significantly simplified.

Using the tag control scheme we can overlap the loading phase for the next sequence with the computing phase of the current sequence as shown in fig.1. It can be seen that the last two elements of the current sequence and the first two elements of the next sequence are overlapped.

The pre-processing stage computes the auxiliary input sequences $x_C(i)$ and $x_S(i)$ using two multipliers to compute equations (7) and (8). Then we have to use a permutation module and an adder to obtain the desired form for the auxiliary input sequence in equations (11) and (12).

The post-processing stage uses two adders to compute $H_C(0)$ and $H_S(0)$ using the equations (5) and (6). Then we recursively compute the auxiliary output sequences $H_C(k)$ and $H_S(k)$ with one adder and one subtract or using equations (9) and (10). Finally, the post-processing stage computes the output sequence using equation (4) with two adders.

V. CONCLUSION

This paper presents an efficient approach to design a systolic array VLSI implementation for a prime length type III generalized discrete Hartley transform based on an appropriate hardware algorithm. The proposed VLSI algorithm is based on an appropriate decomposition of the type III GDHT into two cyclic convolution structures of the same length and structures. The two computational structures are implemented in parallel on the same VLSI array that had been obtained using an appropriate hardware sharing technique and a VLSI architecture paradigm based on using small bi-port ROMs. The obtained structure leads to a high computing speed with a low hardware complexity and low I/O costs together with a high degree of regularity and modularity and a good topology with local connections. The obtained structure is well suited to a VLSI implementation using a memory-based VLSI structure.

REFERENCES

[1] N. C. Hu, H.I. Chang, O. Orsoy, "Generalized Discrete Hartley Transform," *IEEE Trans. Signal Processing*, pp. 2931-2940, vol. 40, no. 6, June 1992.

[2] R. E. Crochiere and I. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Clifs, N.J. Prentice Hall, 1983.

[3] C.M. Rader, "Discrete Fourier Transform when the number of the data samples is prime," *Proc. IEEE*, 1107-1108,vol.56, 1968.

[4] J. Guo, C.M. Liu, C.W.Jen, "A New Array Architecture for Prime-Length Discrete Cosine Transform," *IEEE Trans. on Signal Processing*, pp.436-442, vol.41, no.1, Jan. 1993.

[5] D.F. Chiper, "Novel Systolic Array Design for Discrete Cosine Transform with High Throughput Rate," *Proc. IEEE Symp. on Circuits and Systems*, Atlanta, Georgia, 1996, pp. 746-749.

[6] Y. H. Chan and W. C. Siu, " On the Realization of Discrete Cosine Transform using the Distributed Arithmetic," *IEEE Trans. on Circuits and Systems-Part I*, vol. 39, no. 9, pp. 705-712, Sept. 1992.

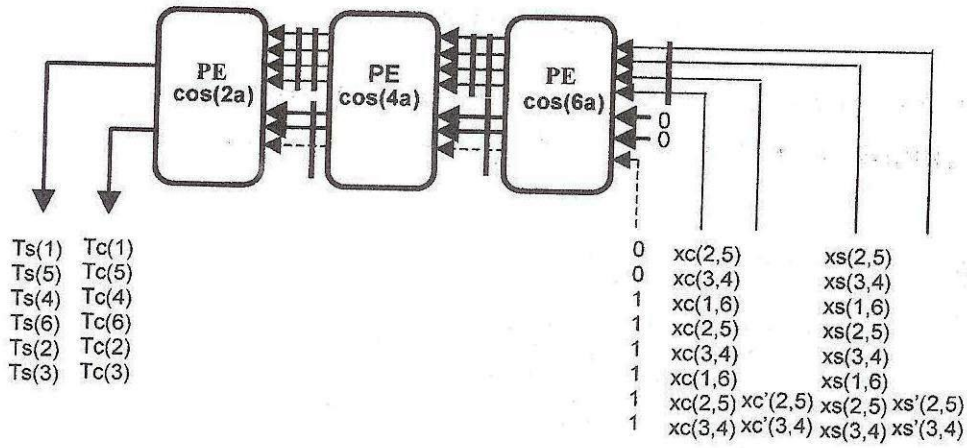
[7] H. T. Kung, "Why Systolic Architectures," *Computer*, 15, Jan. 1982.

[8]S. A. White, "Applications of the Distributed Arithmetic to Digital Signal Processing: A tutorial,"*IEEE ASSP Mag.*, vol.6, July 1989.

[9] S.Y. Kung, "*VLSI Array Processors*," Prentice Hall, Englewood Cliffs, New Jersay, 1988.

[10] C.W. Jen and H.S. Hsu, "The design of systolic arrays with tag input," in *Proc. 1988 IEEE Int. Symp. on Circuits and Systems*, pp. 2263-2266, 1988.

[11] M.T.Sun, et al., "VLSI implementation of 16x16 discrete cosine transform," *IEEE Trans. on Circuits and Systems*, vol.36, no.4,pp.610-617, April 1989.



Note: 1. $a = 2\alpha$

2. $xc(i,j) = x_C(i) + x_C(j)$

and $xs(i,j) = x_S(i) + x_S(j)$

Fig. 1. VLSI architecture for the hardware core of type III GDHT of length N=7

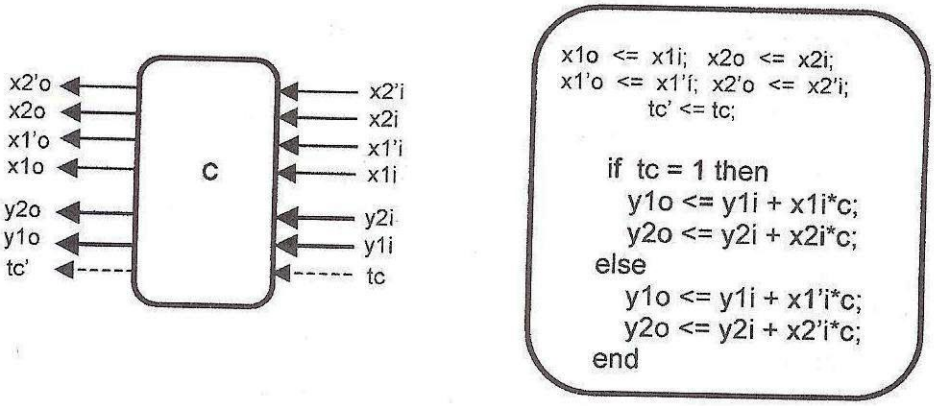


Fig. 2. Functionality of a processing element PE in Fig. 1.