A Novel Architecture for Walsh Hadamard Transforms Using Distributed Arithmetic Principles

A. Amira, A. Bouridane and P. Milligan
School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN, Northern Ireland
(A.Abbes, A.Bouridane, P.Milligan)@qub.ac.uk

ABSTRACT: The Walsh-Hadamard transforms are important in many image processing applications including compression, filtering and code design. A novel architecture for the Fast Hadamard Transform, using distributed arithmetic techniques, is proposed. In the paper, the mathematical model for the algorithm proposed, the associated design using both a distributed arithmetic ROM and Accumulator structure and a sparse matrix factorisation technique are described. The design has $O(2W)$ computation time complexity, where W is the input wordlength, requires less area when compared with existing systolic architectures and is suitable for FPGA implementations.

1. INTRODUCTION

Transform methods are useful in many types of applications, particularly if the features of interest can be characterised in the transform domain. A useful transform in speech and image processing is the Walsh-Hadamard transform (WHT)[4]. It is an orthogonal transform, with only additions and subtractions required, and is faster than sinusoidal-like transforms [2]. It can also be formulated as a matrix-vector multiplication similar to the Discrete Fourier transform DFT, and it has a fast algorithm which has $Nlog_2N$ additions and subtractions for N-point input samples [3],[5]. The Fast Walsh Hadamard Transform (FHT) is similar to the DFT and is globally recursive and requires global communication as required for the shuffling between different stages of the process. A chip for a systolic array of the Hadamard transform was designed by the University of South California and fabricated by MOSIS [3]. Its computation requires $2N$-1 clock cycles while its latency is N cycles. However, the addition implemented in the array is at word level. Thus, the time required is proportional to log_2N+W, where W is the wordlength of the input sample. To increase the speed a new bit level systolic architecture has been developed and reported in [2]. This has a computation time of $(2N-1)(W+log_2N)$ clock cycles with a latency of $N(W+log_2N)$ cycles.

Distributed Arithmetic (DA) distributes arithmetic operations rather than grouping them as multipliers do. Conventional DA, called ROM-based DA, decomposes the variable input of the inner product to bit level in order to generate precomputed data. ROM-based DA uses a ROM table to store the precomputed data, which makes it regular and efficient in the use of silicon area, in a VLSI implementation [6],[7]. However, when the size of the inner products increases the ROM area increases exponentially and becomes impracticably large, even when using ROM partitions [1]. The advantage of a DA based ROM approach is its efficiency of implementation. The basic operations required are a sequence of ROMs, addition, subtraction and shift operations of the input data sequence. All of these functions efficiently map to a field programmable gate array FPGA structures [8].

It is the aim of this paper to present the development of an efficient architecture ideally suited for a fast computation of the FHT. Sparse matrix factorisation methods and the symmetry of the Hadamard matrices are exploited to develop the mathematical model in order to reduce the ROM size, the area consumed by the design, and to speed up the computation procedure by minimising the number of addition and subtraction operations required.

The composition of the rest of the paper is as follows. The mathematical model for the FHT algorithm is given in section 2. Section 3 is concerned with the implementation strategy used in conjunction with DA and sparse matrix factorisation methods. Section 4 describes the architecture produced. Concluding remarks are given in section 5.

0-7803-6542-9/00/$10.00 © 2000 IEEE

2. MATHEMATICAL BACKGROUND

Typically, a Hadamard matrix is defined iteratively as:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}.$$

where H_N is a Hadamard matrix of size NxN. Furthermore, if the transform length N is a power of two (i.e., $N = 2^n$), then a Fast Hadamard Transform (FHT) algorithm (similar to Fast Fourier Transform (FFT)) can be used for its fast computation. Let the input data and the transformed data be represented by the two vectors X and Y of size N, respectively. Then Y can be written as follows:

$$Y = H_N X \tag{1}$$

such that

$$Y_i = \sum_{k=0}^{N-1} H_{N,ik} X_k \tag{2}$$

where $\{X_k\}$'s are written in the fractional format as shown in equation (3)

$$X_k = -x_{k,W-1} + \sum_{m=1}^{W-1} x_{k,W-1-m} 2^{-m} \tag{3}$$

where $x_{k,m}$ is mth bit of x_k (which are zero or one) and $x_{k,W-1}$ are the sign bits, where W is the wordlength, respectively.

Substituting (3) in (2):

$$\begin{aligned} Y_i &= \sum_{k=0}^{N-1} H_{N,ik} \left(-x_{k,W-1} + \sum_{m=1}^{W-1} x_{k,W-1-m} 2^{-m} \right) \\ &= -\sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1} + \sum_{m=1}^{W-1} \left(\sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1-m} \right) 2^{-m} \end{aligned} \tag{4}$$

Define $H_{W-1-m} = \sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1-m} \quad (m \neq 0)$

and $$H_{W-1} = -\sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1} \tag{5}$$

The output result is given by:

$$Y_i = \sum_{m=0}^{W-1} H_{W-1-m} 2^{-m} \tag{6}$$

Since the term H_m depends on the $x_{k,m}$ values and has only 2^N possible values, it is possible to precompute and store these values in a ROM. An input set of N bits $(x_{1,m}, x_{2,m}, \ldots\ldots x_{N,m})$ is used as an address to retrieve the corresponding H_m values.

3. 1-D FHT IMPLEMENTATION

A direct implementation of equation (1) would require (in the case of N=8 and W=12) 56 additions and subtractions. However, with the use of the FHT the number of the addition and subtraction operations is reduced to 24 as shown in Table 1.

Table .1. Three Steps for the FHT

Input data	Step (1)	Step (2)	Step (3)
X_1	$T_1 = X_1 + X_2$	$c_1 = T_1 + T_3$	$Y_1 = c_1 + c_5$
X_2	$T_2 = X_1 - X_2$	$c_2 = T_2 + T_4$	$Y_2 = c_2 + c_6$
X_3	$T_3 = X_3 + X_4$	$c_3 = T_1 - T_3$	$Y_3 = c_3 + c_7$
X_4	$T_4 = X_3 - X_4$	$c_4 = T_2 - T_4$	$Y_4 = c_4 + c_8$
X_5	$T_5 = X_5 + X_6$	$c_5 = T_5 + T_7$	$Y_5 = c_1 - c_5$
X_6	$T_6 = X_5 - X_6$	$c_6 = T_6 + T_8$	$Y_6 = c_2 - c_6$
X_7	$T_7 = X_7 + X_8$	$c_7 = T_5 - T_7$	$Y_7 = c_3 - c_7$
X_8	$T_8 = X_7 - X_8$	$c_8 = T_6 - T_8$	$Y_8 = c_4 - c_8$

To reduce the number of arithmetic operations and to speed up the process, the symmetry in the FHT matrix coefficients and a sparse matrix factorisation are exploited as shown in equations (7), (8) and (9).

$$\begin{aligned} Y &= H_N X \\ &= H_N^4 H_N^2 X \\ &= H_N^4 T \end{aligned} \tag{7}$$

where H_N^4 and H_N^2 are uniformly sparse with 4 and 2 non-zero coefficients respectively.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \\ Y_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \end{bmatrix} \tag{8}$$

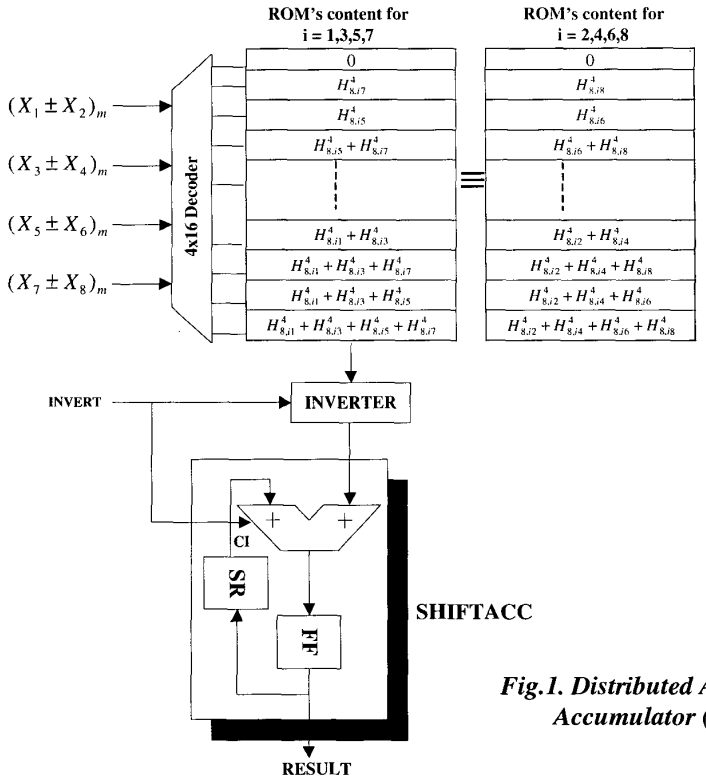

Fig.1. Distributed Arithmetic ROM and Accumulator (RAC) structure

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \\ Y_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} \quad (9)$$

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} = \begin{bmatrix} X_1 + X_2 \\ X_1 - X_2 \\ X_3 + X_4 \\ X_3 - X_4 \\ X_5 + X_6 \\ X_5 - X_6 \\ X_7 + X_8 \\ X_7 - X_8 \end{bmatrix} \quad (10)$$

It can be seen that the number of additions and subtractions which are required in the 2nd and 3rd steps of the FHT computation can be performed using DA principles. Equation (9) is essentially a matrix vector computation, where each element of the output vector Y is formed by multiplying each of the 4 elements of the input vector by one of 4 predefined coefficient values. The key point to note is that the decomposition produces similar coefficients for each two consecutive rows. This is exploited by using the same ROM contents for ROM table (2i+1) and ROM table (2i+2) which are addressed by $(T_{2i+1})_m$ and $(T_{2i+2})_m$, respectively, (i=0,1,2,3). Figure 1 illustrates the structure developed using a conventional ROM and Accumulator (RAC) structure showing the ROM content tables for each value i (i=0,1,2,3,...,8) as calculated from equations (4) and (5) and H_8^4 as described above.

4. 1-D FHT ARCHITECTURE

The architecture for the 1-D FHT is shown in Figure 2. The 12 bit inputs to the circuit are fed in a bit-serial fashion from the converter. The 4 separate RAC blocks calculate the 8 transforms as follows: a butterfly structure of bit-serial adders and subtracters is used to generate the elements of the input matrix in equation (9).

During the first 12 cycles eight bit-parallel outputs for each odd result are produced, and during the second 12 cycles eight bit-parallel outputs for each even results are produced. This implies a bit-parallel data organization with the maximum value

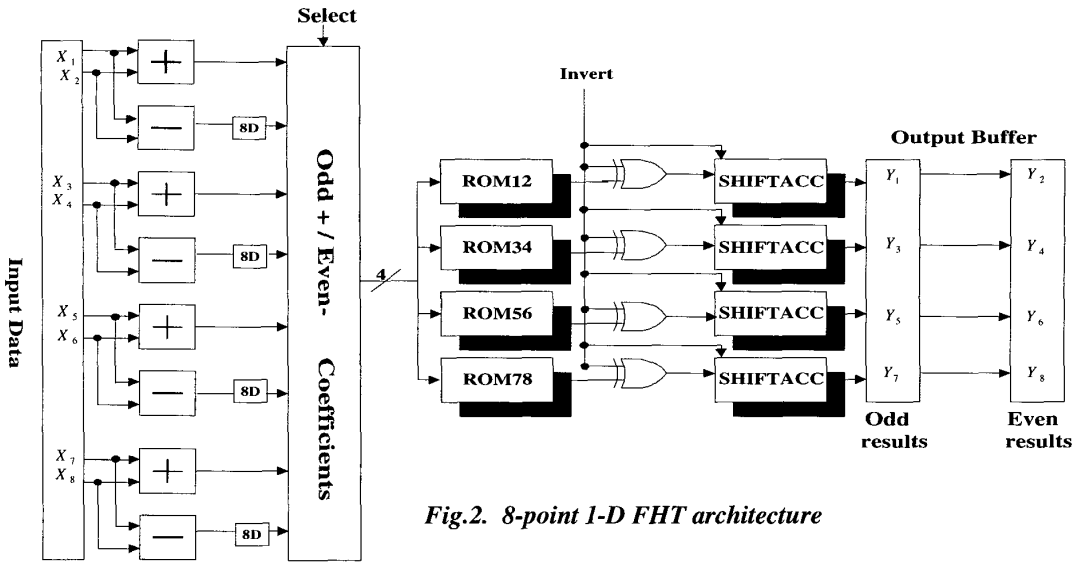

Fig.2. 8-point 1-D FHT architecture

in the ROM represented by 5 bits including the sign extension. Every W clock cycles the signal INVERT is used to compute the two's complement of the ROM's content by inverting the value and inserting a C_{in} (carry in of the adder) of value one while signal SELECT is used to select the odd and even input samples. The selector is basically a multiplexer (8→4), and the computation process runs from $m=0$ to $m = 2\times(W-1)$. The architecture is suitable for FGPA implementation, but it should be noted that the strategy used to implement equation (9) requires a dedicated routing due to the "butterfly" structure of adders and subtracters (see Figure 2) needed to generate the input vector. While the routing can be seen as costly in FPGA structures, the considerable hardware savings achieved by using the DA method, offsets the small increase in hardware and routing complexity required by the butterfly structure.

5. CONCLUSION

The paper has presented a novel architecture for the FHT computation. The underlying mathematical model has been presented and verified during the design and implementation stages. The proposed architecture is suitable for FPGA implementation, modular and requires less area with lower time complexity when compared with the existing structures of [2] and [3].

6. REFRENCES

[1] Keshab K. Parhi," VLSI digital signal processing systems design and implementation." *John Wiley & Sons, Inc. USA,* 1999.

[2] Long-Wen Chang and Ming-Chang Wu, "A bit level systolic array for Walsh-Hadamard transforms." *Signal Processing Vol 31, pp 341-347,* 1993

[3] S.Y.KUNG,"VLSI Array Processors." *PRENTICE HALL, USA*, 1988.

[4] D. Coppersmith et al "Hadamard transforms on multiply/add architecture," *IEEE Trans. Signal Processing, Vol 42, N° 4,, pp. 969-970,* April 1994

[5] M.H. Lee and M.Kaveh, "Fast Hadamard transform based on a simple matrix factorization." *IEEE Trans. Acoust. Speech vol. ASSP-34, no. 6, pp. 1666-1667*, 1986.

[6] K. P. Lim and A.B. Premkumar, "A Modular Approach to the Computation of Convolution Sum Using Distributed Arithmetic Principles." *IEEE Trans. Circuits and Systems -II: Analog and Digital Signal Processing. Vol.46. N°.1.* January 1999.

[7] T-S.Chang et al "New distributed arithmetic algorithm and its application to IDCT." *IEE Proc on Circuit Devices. Syst, Vol 146. N°4,* August 1999.

[8] "The Role of the FPGA- based Signal Processing" *URL: http://www.xilinx.com.*