AN FPGA BASED PARAMETRISABLE SYSTEM FOR DISCRETE ORTHOGONAL TRANSFORMS IMPLEMENTATION

*A.Amira, A.Bouridane, M.Roula and F.Kurugollu

School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN, United Kingdom
*A.Abbes@qub.ac.uk

ABSTRACT

This paper presents novel architectures for efficient implementation of Discrete Orthogonal Transforms (DOTs) using an FPGA based parameterisable system. These transforms are important in many signal and image processing applications including image and speech compression, filtering and coding. Two novel architectures for DOTs using both systolic architecture and distributed arithmetic design methodologies are presented. The first approach uses the Modified Booth-encoder-Wallace trees Multiplication (MBWM) algorithm for a systolic architecture implementation. The second approach is based on both distributed arithmetic ROM and accumulator structure, and Offset Binary Coding technique (OBC). Implementations of the algorithms on a Xilinx FPGA board are described. Distributed arithmetic approach exhibits better performances when compared with the systolic architecture approach.

1. INTRODUCTION

Transform methods are useful in many types of applications, particularly if the features of interest can be characterised in the transform domain [1]. DOTs are important in many image and signal processing applications. The DFT, which is central to most DSP applications, is the most popular and the oldest among these DOTs. Discrete Hartley Transform (DHT) belongs to the family of sinusoidal transforms that map temporal or spatial functions into frequency functions. DHT accomplishes this in a manner similar to the better-known Fourier Transform. The significant difference between the Fourier Transform and Hartley's alternative is that the Hartley transform uses only real values, i.e., no complex numbers [2]. Among all the sinusoidal transforms, the DCT is the most efficient transform for compression of speech and image data. Also, it is well known from the literature that the Fast Hadamard Transform (FHT) belongs to the rectangular transforms and can be efficiently used for the calculation of the DFT for implementing adaptive filters and spectrum filter realisations. The usual frequency -domain FIR filtering problem can easily be converted into a Walsh frequency domain-filtering problem, and a similar structure results in possible alternative for infinite-impulse response filter implementations. The advantages of the 2-D FHT, which is based on 1-D FHT also known as S or sequential transform in lossless image compression, are well known. All the transforms previously described are based on matrix-vector multiplication. The SVD, which is an eigen-vector based matrix-matrix multiplication transform, is used in speech compression and image enhancement [3], [4],[5], [6].
It is the aim of this paper to develop efficient architectures, ideally suited for a fast computation of the DOTs using an FPGA based parameterisable system. Systolic Architecture (SA) and Distributed Arithmetic (DA) design methodologies have been described for the implementation of DOTs. The MBWM algorithm has been used for the implementation of the systolic architecture, due to it suitability for a Virtex FPGA implementation [7]. OBC technique has been exploited to develop the mathematical model in order to reduce the ROM size, the area required by the design, and to speed up the computation time in the case of DA design methodology.
The architectures proposed in this paper have been designed and targeted to the Xilinx XCV1000E of the Virtex-E family which has the following important features [7]:

- Fast, and high-density Field-Programmable Gate Array;
- Flexible architecture that balances speed and density; and
- Built-in clock- management circuitry.

The composition of the rest of the paper is as follows. The system architecture for designing and implementing the DOTs is presented in section 2. The mathematical model for the DOTs algorithm is given in section 3. Section 4 is concerned with the proposed architectures using both SA and DA techniques. The analysis of the implementation results obtained is given in section 5. Concluding remarks are given in section 6.

2. SYSTEM ARCHITECTURE FOR DESIGNING AND IMPLEMENTING THE PROPOSED ARCHITECTURES

The proposed system architecture for designing and implementing the DOTs as shown in Fig.1 consists of:

i) A GUI for supporting experimentation with different design parameters (transform length, input data word length, result word length, coefficients word length) and methodologies (SA, DA) to enable the user to explore e.g. speed/area trade-offs;

ii) The library of architectures for DOTs including DCT, DHT … FHT. The application user has the ability to choose and download existing files from the Standard Template Library (STL), to generate new files and to save those files either to the STL or another application directory;

iii) A generator, which automatically produces VHDL code given the user, selected parameters and settings;

iv) A mechanism for making use of the standard (Xilinx) synthesis tools; and

v) A coprocessor which is based on the Xilinx XCV1000E of the Virtex-E family.

The objectives of the work presented in this paper are as follows:

i) Using FPGA as low cost accelerator for DOTs implementation;

ii) Using FPGA to build systems with advantages over conventional technology such as ASICs, DSPs and CPUs in any of the areas of: Time-to-market, performance, power, size/ weight, flexibility and life cycle cost;

iii) Developing a library of DOTs where this library can be extended for a range of other transforms, the library should be more extensive than existing systems;

iv) Developing novel, efficient and scalable architectures for DOTs, where both the area and the speed can be estimated for any transform with any specific design parameters and methodologies;

v) Enabling application users to concentrate on experimenting conveniently with different transforms and techniques to investigate best area/speed trade-offs, rather than concentrating on the low level (and complex) structure of FPGAs; and

vi) Exploiting the potential of more recent FPGA devices (e.g. Virtex-E).

3. MATHEMATICAL MODEL

Typically, a DOT algorithm is defined as matrix-vector multiplication:

$$Y_i = \sum_{k=0}^{N-1} A_{ik} X_k \qquad (1)$$

where $A=[A_{ik}]$ is the kernel matrix of an orthogonal transform, $X=[X_k]$ is the input vector data, $Y=[Y_i]$ is the result vector and N is the transform length. This problem can be resolved following the proposed design methodologies:

3.1. DOTs based Systolic Architecture (SA)

If the elements of the input vector X is represented using the 2's complement number representation, then:

$$X_k = -x_{k,W-1}2^{W-1} + \sum_{m=0}^{W-2} x_{k,m}2^m \qquad (2)$$

where $x_{k,m}$ is the mth bit of X_k, (which are zero or one). $x_{k,W-1}$ is the sign bit, where W is the word length.

Equation (2) can be rewritten as follows:

$$X_k = \sum_{m=0}^{(W/2)-1} (x_{k,2m-1} + x_{k,2m} - 2x_{k,2m+1})\, 4^m \qquad (3)$$

or:

$$X_k = \sum_{m=0}^{(W/2)-1} (D_m)\, 4^m \qquad (4)$$

where: $x_{k,-1} = 0$ and $D_m \in \{-2,-1,0,1,2\}$

By substituting (4) into (1), the output vector Y_i can be computed as follows:

$$Y_i = \sum_{k=0}^{N-1} A_{ik} \sum_{m=0}^{(W/2)-1} (D_m)\, 4^m \qquad (5)$$

such as:

$$Y_i = \sum_{k=0}^{N-1} \sum_{m=0}^{(W/2)-1} (A_{ik}\, D_m)\, 4^m \qquad (6)$$

The results Y_i are given by equation (7) as follows:

$$Y_i = \sum_{k=0}^{N-1} \sum_{m=0}^{(W/2)-1} (PP_{ik,m})\, 4^m \qquad (7)$$

where : $PP_{ik,m} = (A_{ik}\, D_m)$, based on the modified Booth encoder algorithm as explained in Table 1.

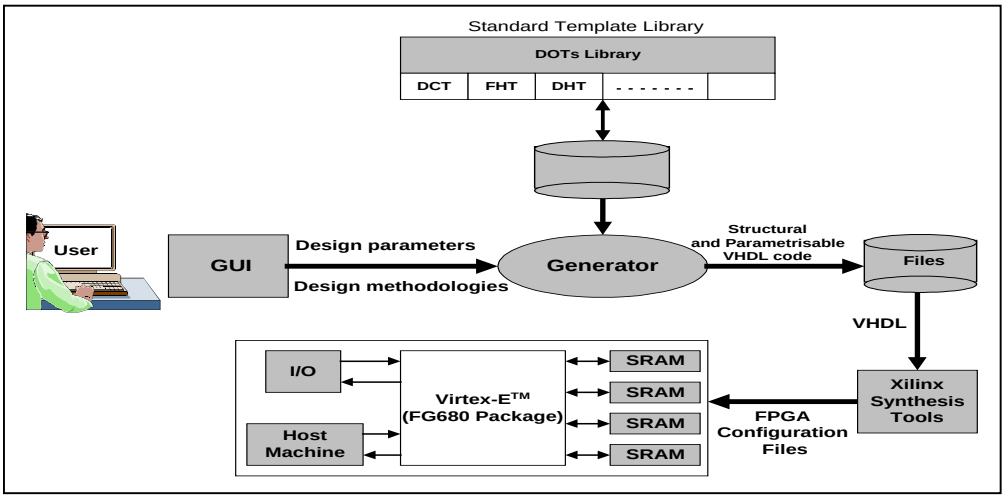

Fig 1. Proposed System Architecture for DOTs Implementation

$x_{k,2m+1}$	$x_{k,2m}$	$x_{k,2m-1}$	D_m	$PP_{ik,m}$
0	0	0	0	0
0	0	1	1	$+A_{ik}$
0	1	0	1	$+A_{ik}$
0	1	1	2	$2A_{ik}$
1	0	0	-2	$-2A_{ik}$
1	0	1	-1	$-A_{ik}$
1	1	0	-1	$-A_{ik}$
1	1	1	0	0

Table 1. Modified Booth encoder algorithm

3.2. DOTs based Distributed Arithmetic (DA)

The OBC technique is introduced to reduce the ROM size by a factor of *2* to 2^{N-1} when using DA principles:
Suppose that $\{A_{ik}\}$'s are L -bits constants and $\{X_k\}$'s are written in the fractional format as shown in equation (2).
Rewrite (2) as:

$$\begin{aligned} X_k &= \frac{1}{2}[X_k - (-X_k)] \\ &= \frac{1}{2}\left[-\left(x_{k,W-1} - \overline{x_{k,W-1}}\right) + \sum_{m=1}^{W-1}\left(x_{k,W-1-m} - \overline{x_{k,W-1-m}}\right)2^{-m} - 2^{-(W-1)}\right] \end{aligned} \quad (8)$$

where

$$-X_k = -\overline{x_{k,W-1}} + \sum_{m=1}^{W-1}\overline{x_{k,W-1-m}}\,2^{-m} + 2^{-(W-1)} \quad (9)$$

Define

$$d_{kj,m} = \begin{cases} x_{k,m} - \overline{x_{k,m}}, & for\ m \neq W-1 \\ -\left(x_{k,W-1} - \overline{x_{k,W-1}}\right), & for\ m = W-1 \end{cases} \quad (10)$$

And $d_{k,m} \in \{-1,+1\}$. Equation (8) can be rewritten as:

$$X_k = \frac{1}{2}\left[\sum_{m=0}^{W-1} d_{k,W-1-m}\,2^{-m} - 2^{-(W-1)}\right] \quad (11)$$

Using (11), (1) can be written as:

$$\begin{aligned} Y_i &= \sum_{k=0}^{N-1}\frac{1}{2}A_{ik}\left[\sum_{m=0}^{W-1} d_{k,W-1-m}\,2^{-m} - 2^{-(W-1)}\right] \\ &= \sum_{m=0}^{W-1}\left(\sum_{k=0}^{N-1}\frac{1}{2}A_{ik}\,d_{k,W-1-m}\right)2^{-m} - \left(\frac{1}{2}\sum_{k=0}^{N-1}A_{ik}\right)2^{-(W-1)} \end{aligned} \quad (12)$$

Now define

$$D_{im} = \sum_{k=0}^{N-1}\frac{1}{2}A_{ik}\,d_{k,m}, \quad for \quad 0 \le m \le W-1 \quad (13)$$

And

$$D_{i\,extra} = -\frac{1}{2}\sum_{k=0}^{N-1}A_{ik} \quad (14)$$

Therefore, Y_i can be computed as:

$$Y_i = \sum_{m=0}^{W-1} D_{i,\,W-1-m}\,2^{-m} + D_{i\,extra}\,2^{-(W-1)} \quad (15)$$

Equations (13) and (14) characterise the OBC scheme. Table 2 shows the content of each the ROM for the case of *N=3*.
It is obvious that the D_{im} values are mirrored along the line between the 8th and the 9th rows in the ROM table. In other words, the term D_m has only 2^{N-1} possible values depending on the $x_{i,m}$ values. Therefore it is possible to reduce the ROM size by a factor of 2. Table 3 illustrates the new ROM table.

$x_{1,m}$	$x_{2,m}$	$x_{3,m}$	**The content of the ROM *i***
0	0	0	$-(A_{i1}+A_{i2}+A_{i3})/2$
0	0	1	$-(A_{i1}+A_{i2}-A_{i3})/2$
0	1	0	$-(A_{i1}-A_{i2}+A_{i3})/2$
0	1	1	$-(A_{i1}-A_{i2}-A_{i3})/2$
1	0	0	$(A_{i1}-A_{i2}-A_{i3})/2$
1	0	1	$(A_{i1}-A_{i2}+A_{i3})/2$
1	1	0	$(A_{i1}+A_{i2}-A_{i3})/2$
1	1	1	$(A_{i1}+A_{i2}+A_{i3})/2$

Table 2. The content of ROM *I*

$x_{1,m}$	$x_{2,m}$	$x_{3,m}$	**The content of the ROM *i***
0	0	0	$-(A_{i1}+A_{i2}+A_{i3})/2$
0	0	1	$-(A_{i1}+A_{i2}-A_{i3})/2$
0	1	0	$-(A_{i1}-A_{i2}+A_{i3})/2$
0	1	1	$-(A_{i1}-A_{i2}-A_{i3})/2$

Table 3. The new content of ROM *I*

4. DOTs ARCHITECTURES

4.1. DOTs based (SA)

The heart of the algorithm given by equation (1) is the multiply-accumulate component. This part takes two values (one row value from the *A* kernel matrix and the column value from the input date vector), multiplies them and adds the result to the running total. Once *N* multiplications and additions have been completed, the result is one value of the *C* matrix. The MAC has been used in this algorithm is basically based on the MBWM as shown in Fig.2 with (*N=3*).
It is worth mentioning that the system produces *N* results *Y* after *O (2N)* clock cycles based on the multiple accumulate technique and requires *N* MACs.

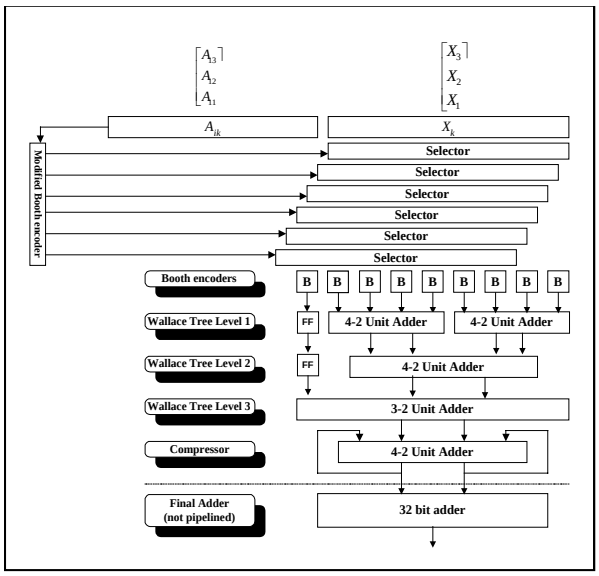

Fig 2. Proposed MAC for DOTs computation

Features	Proposed Structure	Structure of [2]	Structure of [4]
Computation time	*(2N)T*	*(2W)T*	*(2WN)T*
Area Complexity	*O(N)*	$O(N^2)$	*O(N)*

Table 4. Comparison of proposed structure with the existing structures ([2],[4]) for computation of the DOTs

4.2. DOTs based (DA)

Fig.3 shows a typical architecture for the computation of DOTs *(N=3)* using DA principles with OBC scheme. The computation starts from the LSB of x_i , i.e m=0. The XOR gates are used for address decoding, the MUX with the constant D_{iextra} provides the initial value to the shift-accumulator and the XOR gate after the ROM is used to inverse the output of the ROM after W clock cycles. Two control signals S_1 and $S2$ are required, where S_1 is 1 after W clock cycles and 0 otherwise, and S_2 is 1when $m= W$, and 0 otherwise.

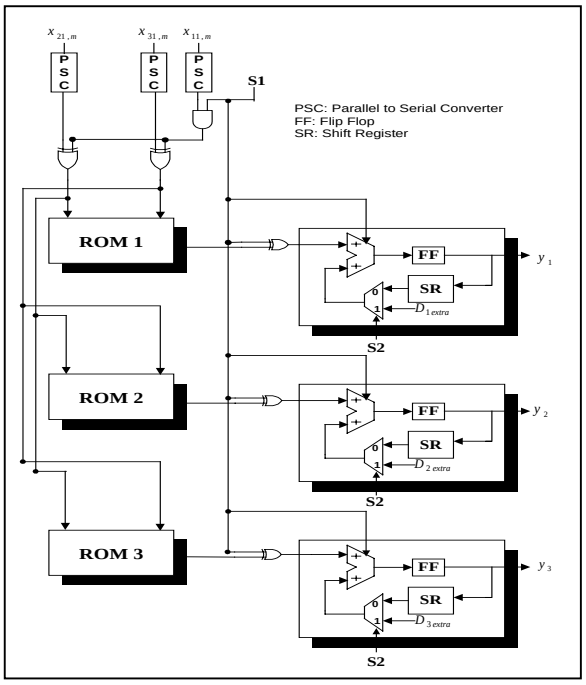

Fig 3. DOTs based OBC using DA Principles (N=3)

Feature	Proposed Structure	Structure of [1]	Structure of [6]
Computation time	*W*	$(2N-1)(W+\log_2 N)$	*2N-1*

Table 5. Comparison of proposed structure with the existing structures ([1],[6]) for computation of DOTs

The input vector elements x_k are fed from the north in parallel/serial fashion while the correspondent's D_m values for each vector $[A_{ik}]$ of the kernel matrix are stored in each ROM.

It is worth mentioning that the result vector produces the three coefficients after W clock cycles.

5. FPGA IMPLEMENTATION

The proposed architectures described above have been implemented for ($N=4$, $W=8$) using a Xilinx Virtex XCV1000E FPGA series board with Target Package: fg680. The designs were carried out using the same Relative LoCations (RLOC) attributes to obtain efficient placement.

The designs are modular, regular and can be implemented for larger transform and input data word lengths. Table.6 illustrates the performance obtained for the proposed architectures in the case of N= 4 and W=8. DA technique shows significant improvements and better performances when is compared with SA technique in terms of speed and area consumed by the design.

Architectures	Slices	Flip-Flops	4-input LUT	Speed (MHz)
SA	540	128	1024	34
DA	90	117	123	122.13

Table 6. Implementation report for SA and DA techniques ($N=4$, $W=8$)

6. CONCLUSION

Due to the importance of the DOTs in image and signal processing applications, a parametrisable system for designing and implementing this kind of transforms has been developed. Two novel architectures have been presented in this paper. The first architecture is based on SA technique while the second one is based on DA principles together with the use of OBC technique. The effectiveness of the two approaches has been discussed and shown that DA approach provides better performances in terms of speed and area when is compared with the SA approach.

7. REFRENCES

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

[2] S.S. Nayak and P.K. Meher, "High throughput VLSI implementation of discrete orthogonal transforms using bit-level vector-matrix multiplier." *IEEE Trans.on Circ.& Syst. II, Analog and Digital Sig. Proc.*, Vol.46, No.5, pp.655-658. 1999.

[3]. A. Amira, A. Bouridane, P. Milligan and M.Roula " An FPGA Implementation of Walsh Hadamard Transforms for Signal Processing." *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, pp 1105 - 1108, Vol: 2, May 7-11, 2001, Salt Lake City, Utah, USA.

[4] A. Amira, A. Bouridane, P. Milligan and P. Sage "A High Throughput FPGA Implementation of A Bit-Level Matrix Product." *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS)*, pp 356-364, October 2000, Lousiana, USA.

[5]. A. Amira, A. Bouridane, P. Milligan and M. Roula "Novel FPGA Implementations of Walsh Hadamard Transforms for Signal Processing." *To be published in the Journal of IEE Proceedings on Vision, Image and Signal Processing.*

[6] S.Y. Kung, "VLSI Array Processors." Prentice Hall, 1988.

[7] URL: www.xilinx.com.