Design and Implementation of a 157 MHz DA-Based DXT CoProcessor

Reza Ebrahimi Atani, Sattar Mirzakuchaki, Farshid Samii, Mohammed Reza Nasrollahzadeh
Iran University of Science and Technology, Narmak, Tehran 16844, Iran

***Abstract* - Frequency analysis using the DFT, the DHT, the DCT or the DST (DTs) is an obvious choice for image and signal processing domain. There are many algorithms for calculating the members of the DT family and each of them has its own merits and weaknesses. In this paper, we have exploited Distributed Arithmetic (DA) to develop a single stand-alone system for calculation of the desired transforms. We have also tried to justify the use of the above-mentioned algorithm. This paper describes the simulation and the implementation of a DXT coprocessor of transform length '8' for the synchronous design in an FPGA device (XILINX VIRTEX-II). The paper presents the trade-offs involved in designing the architecture and the design for performance issues.**

***Index Terms* — DXT, Coprocessor, Distributed Arithmetic.**

I. INTRODUCTION

Memory based Field Programmable Gate Arrays (FPGAs) have the advantage of real-time in-circuit re-configurability as opposed to other gate arrays of similar gate density. This advantage translates into unlimited, in-circuit flexibility, re-configurability and reliability, facilitating prototyping of complex electronic designs [1]. FPGA devices have been used to implement Custom DSPs since the beginning of this decade [2]. Usually, FPGAs are used as VLSI replacement on low volume production or prototyping devices which are to be eventually implemented as ASICs. Their 100% testability and the possibility of achieving a high degree of fault coverage makes them increasingly attractive for complex designs with multiple iterations on their design cycles [1].

The FPGA devices have benefited from the improvements in VLSI technology, leading to higher speed and capability as well as lower power consumption [2]. Discrete Transforms (DT's) has found wide application in signal processing in general and special uses that are customized for a particular task. The DT algorithms are very well known and due to their versatility and very simple hardware implementation are widely used for VLSI digital signal processing systems. The discrete Hartley transform (DHT) has been established as a potential tool for signal processing and communication applications, e.g., computation of circular convolution, and deconvolution, interpolation of real-valued signals, image compression, error control coding, adaptive filtering, multi-carrier modulation and many other applications [3]. DHT has found popularity in recent years and is expected to replace DFT in many spectral analysis schemes. Discrete Fourier Transform (DFT) is the foundation of the DT family and there are numerous algorithms for its computation. DFT is especially good for spectral analysis but has some other uses such as image processing, solving partial differential equations and multiplication of large integers. DFT and DHT are also very important because they can be used to compute the convolution of two signals. Discrete Cosine Transform (DCT) has long been used in image and speech processing and is the optimum fast algorithm for image and data compression applications and forms a key role in many image and video compression standards including JPEG2000 for still image compression, ITU H.261 and H.263 in teleconferencing and ISO MPEG1 and MPEG2 for moving pictures and home video. In addition DCT has been used in filtering and feature extraction [4, 5, 6]. The discrete sine transform (DST) is useful for spectrum analysis, data compression, speech processing, biomedical signal processing and in many other applications. These basic signal processing transforms are required in almost all the phases of image and signal processing and cover a large range of biomedical signal and image processing, for various imaging techniques and spectral analysis of the signals [5]. There are various algorithms for computation of Discrete Transforms that have been developed over the years. In the case of DFT we have Radix-2, Radix-4, Split-Radix, Fast Hartley Transform (FHT), and the Decimation-in-Time-Frequency (DITF) algorithms to name just a few. A number of architectures are proposed for the realization of these transforms [2-8]. However, a unified architecture, which can compute all these transforms, can serve the purpose of a general DSP chip, and therefore a unified architecture has been adopted to obtain all the transforms in a single FPGA chip. There are some implementations for the DXT calculations, but two of the more important ones are Systolic architecture (SA) and Distributed Arithmetic. In this paper we have used DA due to its suitability for FPGA implementation and because it shows dramatic improvements and better performances in comparison with the SA technique in terms of speed and area consumption [6]. The basic structure of all the transforms, DFT, DCT, DHT and

DST, are almost equivalent and this property has been exploited in the design of the unified architecture. In fact we have used DA to calculate DCT and DST transforms and added some glue logic to make the architecture suitable for calculation of the other two transforms i.e. DHT and DFT.

II. DISTRIBUTED ARITHMETIC

Distributed Arithmetic (DA) plays a key role in embedding DSP functions in FPGA devices because the algorithm is based on Look-up table (LUT). DA is one of the most common techniques where the multiply – accumulate is of paramount importance. Its greatest advantage is in using additions instead of multiplications which is desirable since a multiplication consumes much more time than an addition.

III. DXT TRANSFORMS

This Section presents the transforms in detail and the possibility of their implementation as the basic processing elements. For a real sample sequence f (n), where $n \in (0,1,...,N-1)$ the DXT, a collective term representing the DFT, the DHT, the DCT and the DST, can be defined as:

DCT:

$$C(k) = \sqrt{\frac{2}{N}} e_k \sum_{n=0}^{N-1} x(n).\cos\left[p\,(2n+1)k/2N\right]$$

$$k = 0,1,2,....N-1 \quad where\ e_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & otherwise \end{cases} \qquad (1)$$

DST:

$$S(k) = \sqrt{\frac{2}{N}} p_k \sum_{n=0}^{N-1} x(n).\sin\left[p\,(2n+1)k/2N\right]$$

$$k = 1,2,....N \quad where\ p_k = \begin{cases} \frac{1}{\sqrt{2}} & k = N \\ 1 & otherwise \end{cases} \qquad (2)$$

DFT

$$F(k) = \sum_{n=0}^{N-1} x(n)\left[\cos\left(\frac{2pkn}{N}\right) - j\sin\left(\frac{2pkn}{N}\right)\right]$$

$$k = 0,1,2,\mathbf{L},N-1$$

$$F(k) = F_x(k) + jF_y(k) \qquad (3)$$

DHT

$$H(k) = \sum_{n=0}^{N-1} x(n)\left[\cos\left(\frac{2pkn}{N}\right) + \sin\left(\frac{2pkn}{N}\right)\right]$$

$$k = 0,1,2,\mathbf{L},N-1 \qquad (4)$$

A. DCT and DST

According to the definition of DCT, for a given data sequence $\{x(n) : n = 0, 1, 2, ..., N-1\}$, the DCT data sequence $\{C(n) : n = 0, 1, 2, ..., N-1\}$ is given by Equation (1).Or in a different representation:

$$C(k) = \sum_{n=0}^{N-1} A_{n,k}(x_n) \qquad (5)$$

Where, A is the transform matrix. The real-time computation of DCT requires a lot of calculations because of the large number of multiplications involved, therefore much effort has gone into reducing the total number of multiplies. From Chen et al [9] it can be shown that we can divide the transform matrix A into two smaller matrices, significantly reducing the total number of multiplications. The DCT is now obtained from the following two equations:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} A_{n,k}(x_n + x_{N-1-n}) \quad \text{for even k} \qquad (6)$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} A_{n,k}(x_n - x_{N-1-n}) \quad \text{for odd k} \qquad (7)$$

In an 8-point DCT transform matrix A in Equation(5) is an (8×8) matrix and due to the symmetry of A it can be replaced by two (4×4) matrices which can be computed in parallel, as can the sums and differences forming the Equations (8,9)

$$\begin{bmatrix} Y_0 \\ Y_2 \\ Y_4 \\ Y_6 \end{bmatrix} = \begin{bmatrix} d & d & d & d \\ b & f & -f & -b \\ d & -d & -d & d \\ f & -b & b & -f \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix} \qquad (8)$$

$$\begin{bmatrix} Y_1 \\ Y_3 \\ Y_5 \\ Y_7 \end{bmatrix} = \begin{bmatrix} a & c & e & g \\ c & -g & -a & -e \\ e & -a & g & c \\ g & -e & c & -a \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \qquad (9)$$

Figure 1 illustrates the overall architecture of an 8-point 1-D DCT. We can implement the 4-product MAC in two ways: one using conventional arithmetic and the other with serial distributed arithmetic as is shown in figures 2 and 3. It is clearly obvious that great reductions on the hardware will be achieved when using the distributed arithmetic. So In our implementation we have used the 4-product MAC using Serial Distributed Arithmetic.

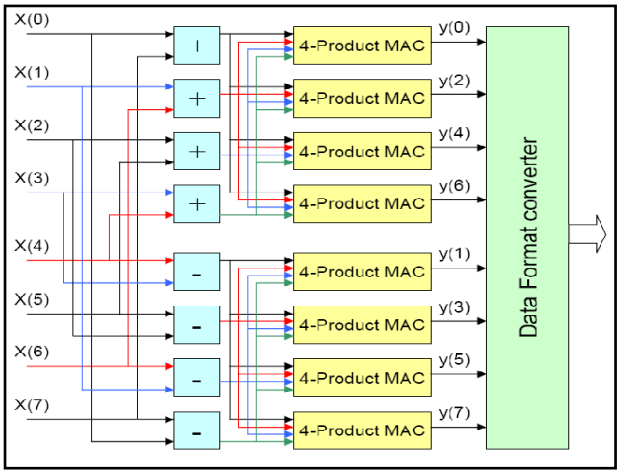

Fig.1. 8-point 1-D DCT.

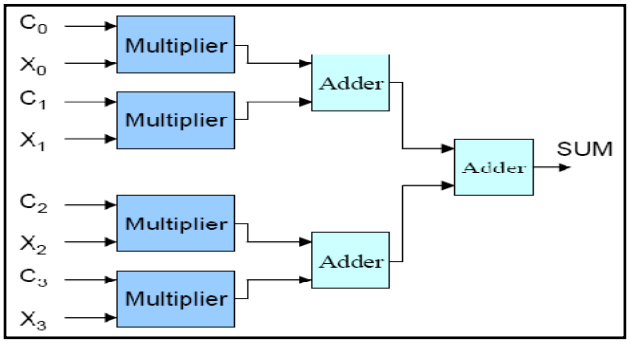

Fig.2. 4-product MAC using Conventional Arithmetic

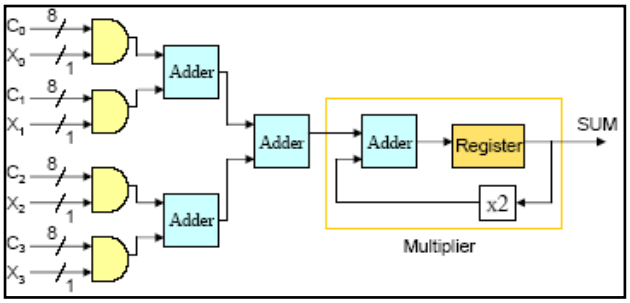

Fig.3. 4-product MAC using Serial Distributed Arithmetic

We now give a brief explanation of how this architecture works using the 8-point DCT example presented in figure 1. The four sum boxes in fig.1 are the equivalents of the corresponding sums in parenthesis in Equation (6) and so are the four difference boxes and differences in Equation (7). In the next stage we have 4-product MACs which each one consists of four multiplication units that are added together to produce the SUM output as illustrated in figure 2 (although, we have used the architecture in figure 3, which produces the same result). If we examine Equations (6, 7) again, we can observe that the coefficient matrices are 4x4 in the case of an 8-point DCT and that is why we have used 4-product MACs. Each 4-product MAC calculates the multiplication of two vectors: one row of the coefficient matrix and the other vector consisted of four sum boxes (or difference boxes). The outputs of the MACs are not in the correct order and need reordering. In the final stage there is a Data Format Converter that has the task of rearranging the outputs of the MACs and delivers the final DCT result.

It should be noted that the calculation of DST is similar to DCT with the only distinction being the different coefficient matrices. So we are not going into details about it any further.

The reduction in hardware in distributed arithmetic is evident from the previous figures. We have used ROMs to store the coefficients needed, speeding up the transform calculation by a large degree. But using ROMs has the disadvantage of adding to the hardware cost, which is undesirable. It can be ameliorated by using the CORDIC algorithm to calculate the coefficients but in real-time applications such as video and image processing it is not feasible and the use of the CORDIC algorithm is left for special CORDIC processors where speed is not vital.

B.DHT and DFT using DCT and DST

In the previous section we developed a system for calculation of DCT (and DST) using DA. Now we want to use this system to compute the other two transforms.

$$X_{DCT}(k) = \sum_{n=0}^{N} x(n) Cos \frac{pnk}{N} \quad k = 0,\ldots, N-1 \qquad (10)$$

The above equation is the definition of an N+1-point discrete cosine transform (DCT). The definition for an N-1 point discrete sine transform (DST) is given below:

$$X_{DST}(k) = \sum_{n=1}^{N-1} x(n) Sin \frac{pnk}{N} \quad k = 1,\ldots, N-1 \qquad (11)$$

There exist the following symmetry relations in the sine and cosine functions:

$$Cos\left(\frac{2p(N-n)k}{N}\right) = Cos\left(\frac{2pnk}{N}\right) \qquad (12)$$

$$Sin\left(\frac{2p(N-n)k}{N}\right) = -Sin\left(\frac{2pnk}{N}\right) \qquad (13)$$

We can divide the input signal into its even and odd parts as follows:

$$x_e(k) = x(k) + x(N-k) \qquad k = 1,\ldots, N/2-1 \qquad (14)$$

$$x_o(k) = x(k) - x(N-k) \qquad k = 1,\ldots, N/2-1 \qquad (15)$$

With the exceptions:

$$x_e(0) = x(0), x_e(N/2) = x(N/2) \qquad (16)$$

We can now define the N-point DFT as:

$$X(k) = X_{DCT}(k) - jX_{DST}(k)$$
$$k = 1,\ldots, N/2 - 1 \quad (17)$$

$$X(N-k) = X_{DCT}(k) + jX_{DST}(k)$$
$$k = 1,\ldots, N/2 - 1 \quad (18)$$

With the special cases:

$$X(0) = X_{DCT}(0), X(N/2) = X_{DCT}(N/2) \quad (19)$$

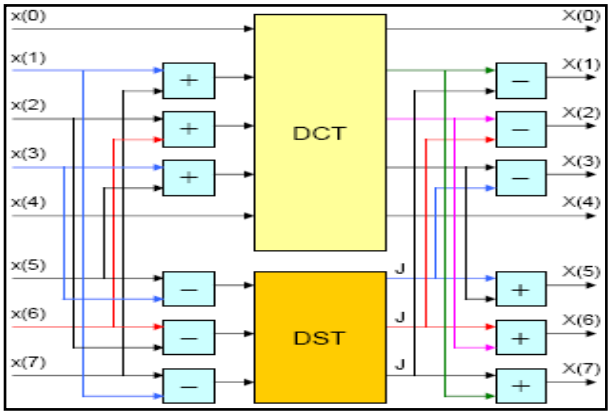

Fig. 4. DFT architecture using DCT and DST

It is now possible to compute the DFT using DCT and DST with the above equations. The calculation of the DHT is the same if with omit the *j* from the Equations (17) and (18). It implies that for computing the N-point DFT (or DHT) we need to compute an $\frac{N}{2}-1$ point DCT and an $\frac{N}{2}-1$ Point DST and then add or subtract them accordingly to achieve the desired results, as is evident graphically from figure 4. Now we have to combine the developed systems into a single general system that is capable of calculation of all four transforms. We have shown the final system that is to be implemented in an FPGA, in figure 6.

VI. IMPLEMENTATION RESULTS

The whole architecture including the computation, data path, and control unit is modeled at Register Transfer Level in VHDL, simulated and tested by a test bench using Modelsim SE 6.1b and implemented in a FPGA device (XILINX VIRTEX-II).

Since we have computed the DFT and DHT using DCT and DST, the total MACs required for each of the transforms as can be seen from Fig. 6 is 8 MACs. The RTL view of the proposed MAC is shown in Fig. 5.

The Hardware description of this architecture for DCT, DST, DHT and DFT implementations of transform length '8' was synthesized using FPGA tools (Synplify VHDL Compiler, version 8.1) and mapped on the xc2v250cs144-6 FPGA chip. In the 8-bit DXT implementation, the estimated frequency is 157 MHz the routed IP takes total of 1777 LUTs which is 57 percent of the chip. Note that this technique can be easily extended to implement 16 bit DXT since the symmetry relations of (12) and (13) are still true and the hardware resources will grow linearly.

VII. CONCLUSION

This paper has proposed an efficient architecture of a common DXT Coprocessor of transform length '8' for the synchronous design. The Direct fast DCT algorithm based on Chen et al's method was presented and then a method of composing the discrete sine-transform from the discrete cosine transform is demonstrated. The DHT is implemented by DFT, which is based on DCT/DST Algorithm. We have exploited Distributed Arithmetic (DA) in order to achieve surface reduction and precision amelioration compared to conventional algorithm.

REFERENCES

[1] J.Davidson "FPGA Implementation of a Reconfigurable Microprocessor", *IEEE Custom Integrated Circuits* Conference p.p.3.2.1 - 3.2.4, 9-12 May 1993.

[2] P.K.Meher, T.Srikanthan, J.C.Patra "Scalable and Modular Memory-Based Systolic Architectures for Discrete Hartley transform", *IEEE Trans. on Circuits and Systems I*: regular papers, VOL. 53, NO. 5, MAY 2006.

[3] A. Amira and A. Bouridane " An FPGA Implementation of Discrete Hartley Transforms", *IEEE Seventh International Symposium on Signal Processing and Its Applications*, Volume 1, P.P.625 - 628, 1-4 July 2003 .

[4] B.Das, S.Banerjee, "Unified CORDIC-based chip to realize DFT/DHT/DCT/DST", *IEE Proc. Computer Digit. Tech.*, Vol. 149, No. 4, July 2002.

[5] K.R Rao and P.Yip "Discrete Cosine Transform, Algorithms, Advantages, applications", Academic Press, San Diego, California, 1990.

[6] M.Amiri, R.Ebrahimi Atani, S.Mirzakuchaki, M.Mahdavi "Design and Implementation of 50 MHz DXT Coprocessor" 10th Euro micro Conference on Digital System Design, Lubeck, Germany, August 29-31, 2007.

[7] Angarita, F.P.Pascual, A.Sansaloni, T.Vails, "Efficient FPGA implementation of CORDIC algorithm for circular and linear coordinates", Field Programmable Logic and Applications International Conference, p.p. 535 – 538, 24-26 Aug. 2005.

[8] H. EL-Bannai, A. A. EL-Fattah, W. Fakhr, "An efficient implementation of the 1D DCT using FPGA technology", *ICM 2003*, Dec. 9-11, Cairo, Egypt, 2003.

[9] W. Chen, C.H.Smith, S.C.Fralick, "A fast computational algorithm for the discrete Cosine transform", *IEEE Trans. Commun.* , VOL. Com-25, p.p. 1004-1009 1977.

[10] B.G. Lee "A new algorithm to compute the discrete cosine transform", *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-32, p.p.1243-1245, Dec. 1984.

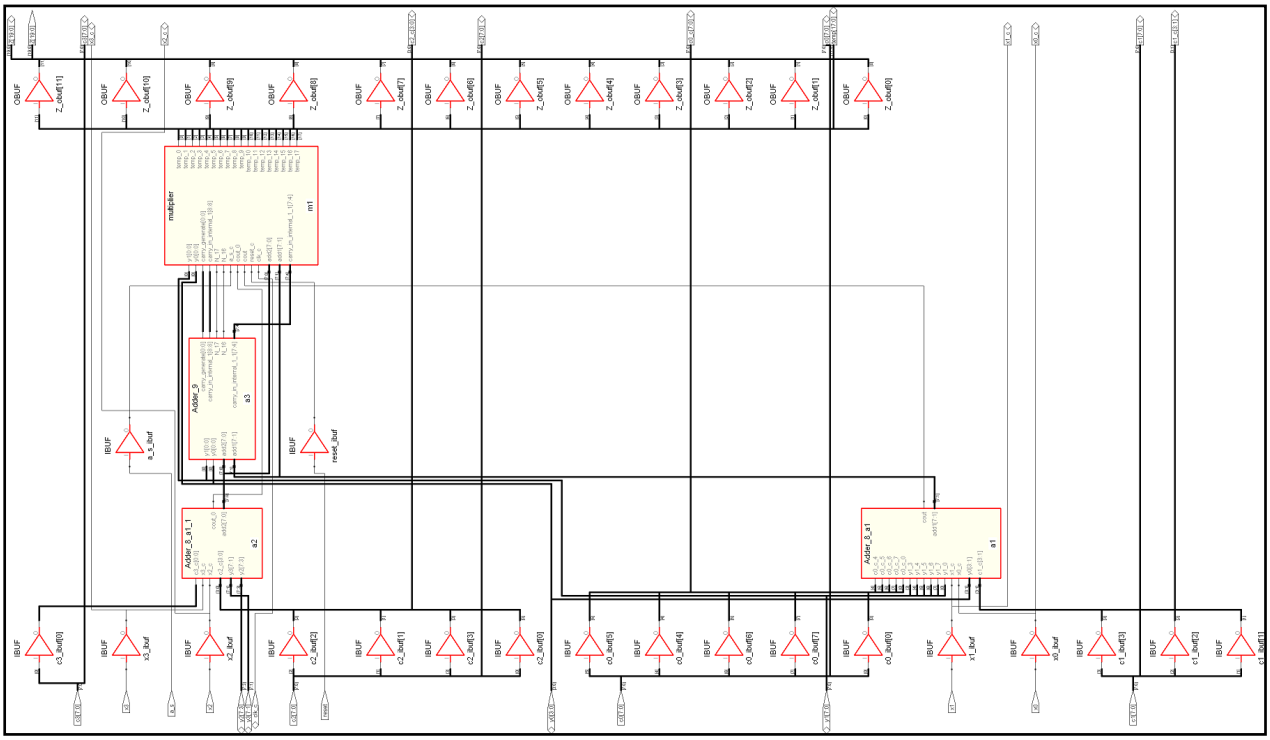

Fig. 5. RTL view of the proposed MAC

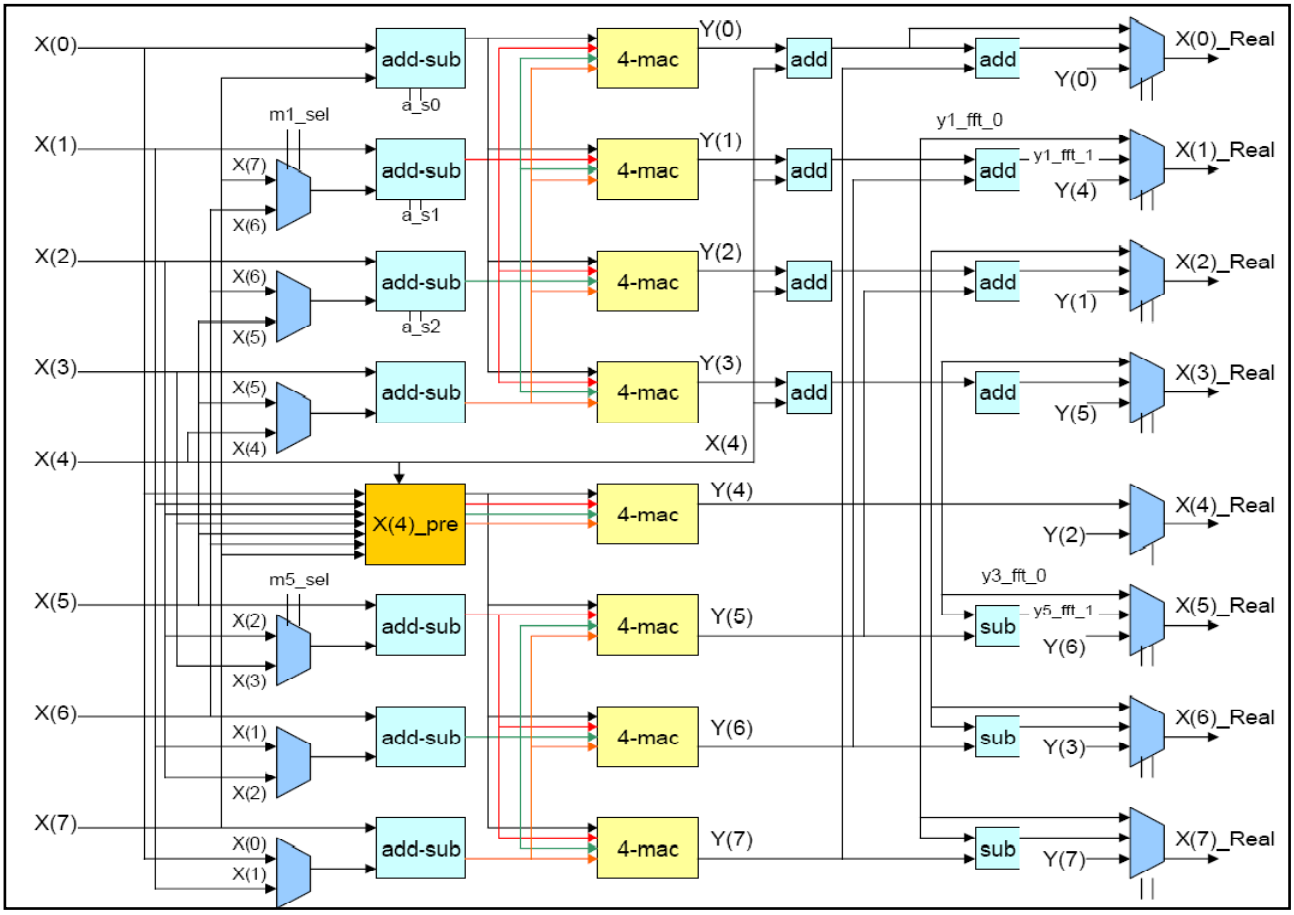

Fig. 6. Final DXT System