A High Performance Pipelined Discrete Hilbert Transform Processor

WANG XU, ZHANG YAN and DING SHUNYING
Department of Electronic and Information Engineering
Shenzhen Graduate School, Harbin Institute of Technology
Shenzhen, Guangdong 518055
China
wangxugo@foxmail.com, ianzh@foxmail.com, http://www.hitsz.edu.cn

Abstract: - A high performance pipelined discrete Hilbert transform (HT) processor is presented in this paper. The processor adopts fast Fourier transform (FFT) algorithm to compute discrete HT. FFT is an effectively method to compute the discrete HT, because the discrete HT can be calculated easily by multiplication with +*j* and -*j* in the frequency domain. The radix-2 FFT algorithm with decimation-in-frequency (DIF) and decimation-in-time (DIT) decomposition are both utilized to construct an efficiently discrete HT signal flow graph (SFG). Some stages in the discrete HT SFG don't include multiplications. These stages are combined into one stage by easy swapping operations to decrease the computational latency. The discrete HT processor is composed of four types pipelined processing elements (PE). Some constant multiplications in these PEs are optimized to reduce the hardware resource. Data being processed is of fixed point mode with 16-bit word width. The pipelined discrete HT processor has the ability to simultaneously perform calculations on the current frame of data, read input data for the next frame of data, and output the results of the previous frame of data. The symmetric property of twiddle factors is utilized to decrease half size of the read-only memory (ROM). Pipelined arithmetic units (adders and multipliers) are designed to enhance the performance of the discrete HT processor. The performance analysis with some previous paper approaches show that the proposed discrete HT processor has the shortest clock latency in discrete HT computation with same samples.

Key-Words: - discrete Hilbert transform, FFT, Adder, Multiplier, FPGA, VLSI

1 Introduction

The discrete HT was developed by Kak, Cizek, and Oppenheim [1][2][3] for applying digital signal processing (DSP) techniques to analytic signal, minimum phase sequence etc. discrete HT is a very important technique in signal and network theory, and have been of practical importance in various DSP systems. Band pass sampling, analytic signal, minimum phase networks and much of spectral analysis theory are based on discrete HT [4].

The most widely used method for computing the discrete HT is through the use of the FFT. Since the early paper by Cooley and Tukey [5], a large number of FFT algorithms have been developed such as radix-2 algorithms, Winograd algorithm (WFTA) [6], prime factor algorithms (FPA) [7], and fast Hartley transform (FHT) [8]. These methods use different transforms to compute the discrete HT, their basic method of computing the discrete HT is that they all use the transform domain for computing the discrete HT. There are other methods, such as the filter method [9] and the systolic arrays [10]. This method comes directly from the discrete HT definition. This method is the direct implementation of the convolution operation on the input sequence with the impulse response of the Hilbert transformer [3]. The filter method requires considerable memory in cases of higher accurate requirement. The systolic arrays method computes the constant parameter matrix beforehand, and then multiplies the input data by this matrix, but the processing unit of the systolic arrays is difficult to implement.

For hardware implementation, architectures of discrete HT processor based on FFT algorithms can be generally grouped into pipelined and memory based architecture styles. Various FFT processors have been proposed in [11]-[21]. The pipelined FFT processors have two popular design styles. One is single-path delay feedback (SDF) pipelined architecture [11] [12], and the other is multi-path delay commutator (MDC) pipelined architecture [13]. Memory based architectures are widely used to design configurable discrete HT processors due to its constant PE and easy memory address management. Memory based architectures usually include one or more PEs, and the hardware cost and the power consumption are both lower than other architectures. Pipelined architectures are good at memory usage, as well as supporting stream data computation. The pipelined architectures are usually used to perform a constant point size discrete HT in high performance environment.

Discrete HT can be computed directly by general FFT processors, but this method is inefficient and has low performance. Discrete HT algorithms based on FFT has some special features which can reduce the latency significantly, so a pipelined specific processor is presented for high speed discrete HT computation. The discrete HT SFG includes two FFTs. Some stages in discrete HT SFG are combined into one stage by easy swapping operations. The discrete HT processor is composed of four types pipelined PEs. The pipelined discrete HT processor has the ability to perform successive frame data in stream mode. All arithmetic units in the processor are working in pipelined mode.

The rest of this paper is organized as follows. In the next section we review the discrete HT definition, and then discuss its related computational methods and algorithms. In section III a novel discrete HT SFG and the architecture of the proposed discrete HT processor is illustrated. Then four types pipelined butterfly PE, multipliers and adders are presented in detail in this section. Performance evaluation and comparison of various discrete HT architectures is presented in section IV. Finally, concluding remarks are given in Section V.

2 Discrete HT and FFT Algorithms

This section gives a brief review on definitions and computational methods of discrete HT. Radix-2 FFT/IFFT algorithm is also discussed.

2.1 The Discrete Hilbert Transform

The HT of signal $x(t)$ is defined as [3]

$$\hat{x}(t)=\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}d\tau = x(t)*\frac{1}{\pi t} \quad (1)$$

where $\hat{x}(t)$ is the Hilbert transform result.

The HT in the frequency domain is given by

$$\hat{X}(\omega)=-j\,\mathrm{sgn}(\omega)X(\omega) \quad (2)$$

Where $X(\omega)$ is the Fourier transform of $x(t)$ and

$$-j\,\mathrm{sgn}(\omega)=\begin{cases}-j & \omega>0\\ +j & \omega<0\end{cases} \quad (3)$$

The discrete HT is developed as an exact equivalent of the Hilbert transform for discrete signals and is defined as

$$\hat{x}(n)=x(n)*h(n) \quad (4)$$

Where $h(n)$ is the impulse of discrete HT given by

$$h(n)=\begin{cases}0 & n=0\\ (1-(-1)^n)/n\pi & n\neq 0\end{cases} \quad (5)$$

The discrete HT can be computed via FFT as shown below

$$\hat{x}(n)=\mathrm{IFFT}(-j\,\mathrm{sgn}(m)X(m)) \quad (6)$$

Where $X(m)=\mathrm{FFT}(x(n))$ and

$$-j\,\mathrm{sgn}(m)=\begin{cases}-j & \mathrm{m}\in[1,N/2-1]\\ 0 & \mathrm{m}=0,\ N/2\\ +j & \mathrm{m}\in[N/2+1,\mathrm{N\text{-}1}]\end{cases} \quad (7)$$

It is evident that the discrete HT can be calculated easily by FFT in three steps. This method transforms the input sequence to the frequency domain, then computes the Hilbert transform in the frequency domain and finally performs an IFFT operation to get the required Hilbert-transformed sequence.

2.2 The Fast Fourier Transform

The discrete Fourier transform (DFT) is the most straightforward mathematical method for finding the frequency content $X(m)$ of a sequence $x(n)$ in the time domain. The *N*-point DFT and Inverse DFT (IDFT) are defined as:

$$X(m)=\sum_{n=0}^{N-1}x(n)W_N^{nm},\ \mathrm{m}\in[0,N-1] \quad (8)$$

$$x(n)=\frac{1}{N}\sum_{m=0}^{N-1}X(m)W_N^{-nm},\ n\in[0,N-1] \quad (9)$$

The twiddle factor $W_N=\exp(-j2\pi/N)$ denotes the *N*-point primitive root of unity. The IDFT can be rewritten as:

$$x(n)=\frac{1}{N}\left(\sum_{m=0}^{N-1}X^*(k)W_N^{nm}\right)^*,\ n=0,1,\ldots,N-1 \quad (10)$$

The equations (8) and (10) have the same twiddle factors and the similar mathematical expression, so DFT and IDFT can be performed by same hardware. N^2 complex multiplications need to be calculated in equations (8) or (10), so a straightforward hardware implementation of the DFT algorithm is obviously impractical. Therefore, the FFT was developed to efficiently speed up DFT computation time and significantly reduce the amount of multiplications.

FFT was proposed by Cooley and Tukey [5] in 1965. FFT is an efficient approach for reducing the computational complexity of DFT. Generally, FFT treats input sequence by decimation-in-frequency (DIF) or decimation-in-time (DIT) decomposition to build a regular SFG.

In radix-2 DIF FFT, $x(n)$ can be segmented into its even and odd indexed elements, then equation (8) is break into two parts as

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n) W_N^{2nm} + W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1) W_N^{2nm} \tag{11}$$

Because $W_N^2 = W_{N/2}$, so

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nm} + W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nm} \tag{12}$$

$$X(m+N/2) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{n(m+N/2)} + W_N^{(m+N/2)} \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{n(m+N/2)} \tag{13}$$

Because $W_{N/2}^{n(m+N/2)} = W_{N/2}^{nm}$ and $W_N^{(m+N/2)} = -W_N^m$, so

$$X(m+N/2) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nm} - W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nm} \tag{14}$$

let

$$A(m) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nm}, B(m) = \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nm}$$

Then equations (12) and (14) are simplified to the form

$$X(m) = A(m) + W_N^m B(m) \tag{15}$$

$$X(m+N/2) = A(m) - W_N^m B(m) \tag{16}$$

For an N-point DFT, we perform an $N/2$-point DFT to get the first $N/2$ outputs and use those to get the last $N/2$ outputs. If N is in powers of two, The DFT can be divided into $\log_2 N$ stages.

2.4 Optimization of Constant Multiplications

Some constant multiplications can be simplified to reduce the chip area in discrete HT. For instance, an input signal multiplied by twiddle factors $W_N^{N/8}$ or

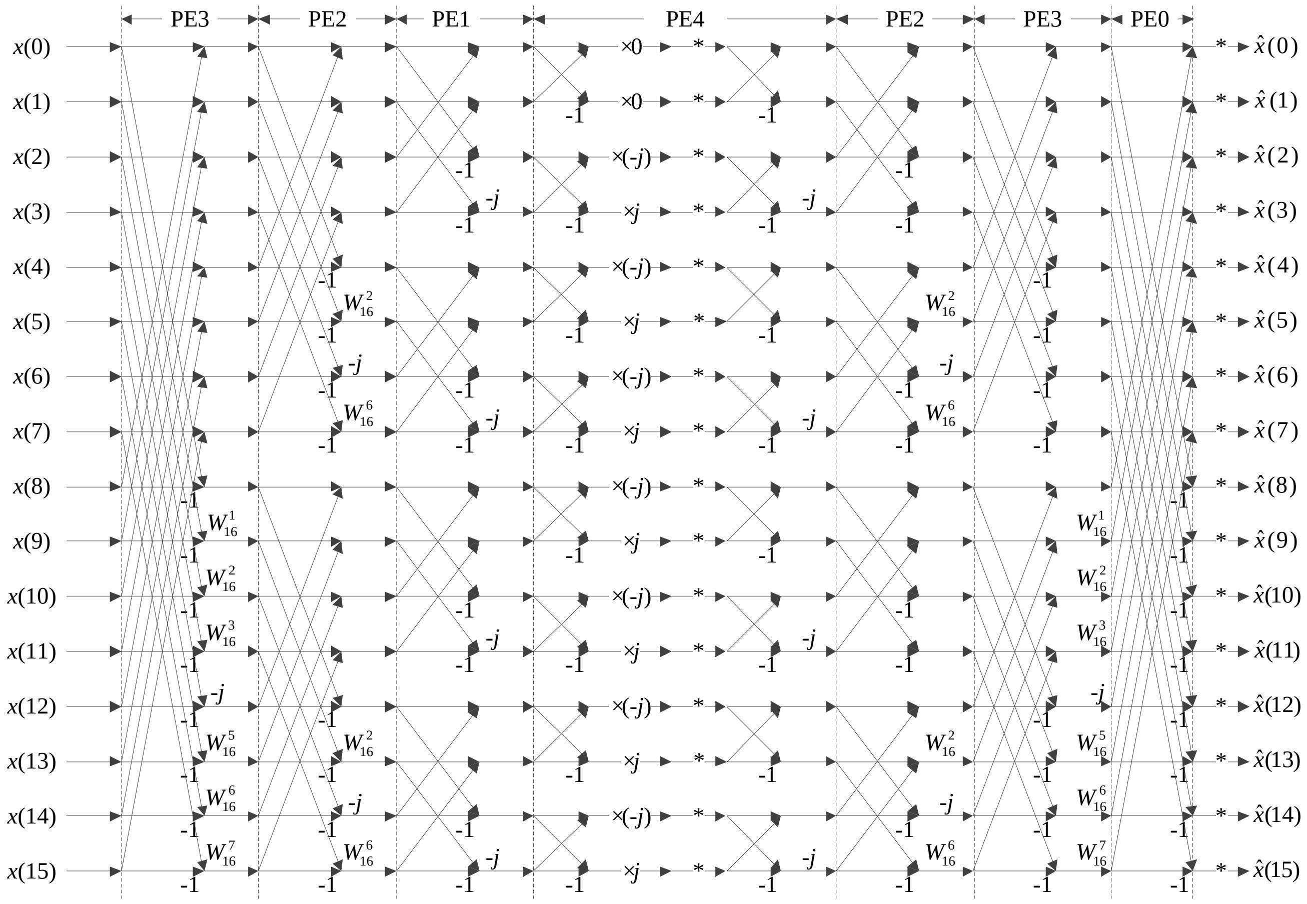

Fig.1. The 16-point discrete HT SFG by two radix-2 DIF FFTs

$W_N^{3N/8}$ can be expressed as:

$$(a+jb)W_N^{N/8} = \sqrt{2}((a+b)+j(b-a))/2 \quad (17)$$

$$(a+jb)W_N^{3N/8} = \sqrt{2}((b-a)-j(a+b))/2 \quad (18)$$

where $a+jb$ denotes a input signal data in complex form. Eqn. 17/18 saves two multipliers, and they need only two adders and two multipliers to perform twiddle factor multiplication.

Another example, data multiplied by $\pm j$ are shown below:

$$j(a+jb) = -b+ja \quad (19)$$

$$-j(a+jb) = b-ja \quad (20)$$

Eqn. 19/20 can be obtained by swapping and negating operation easier than multiplications, so data multiplied by $\pm j$ can be computed without adders and multipliers. The circuit implementation of these constant multipliers will be illustrated in the next section.

3 The Proposed discrete HT SFG and associated Architecture

3.1 The Proposed discrete HT SFG

Discrete HT can be computed by two FFTs. As an example, a 16-point discrete HT SFG is depicted in Fig.1. The discrete HT SFG has seven stages, and these stages are computed by PE0, PE1, PE2, PE3 and PE4. In fact, discrete HT depicted by this SFG form with point size N, where N is in powers of two, can be divided into $2\log_2 N - 1$ stages and performed by these five types of PEs.

The construction of discrete HT SFG needs to avoid bit reversed sorting. Bit reversed sorting is a time consuming operation, and it needs additional memories for swapping data. Two FFTs with different decomposition in the discrete HT SFG can avoids bit reversed sorting. In most cases, natural order data meets our work demands, so FFT in discrete HT SFG is of DIF decomposition, and IFFT

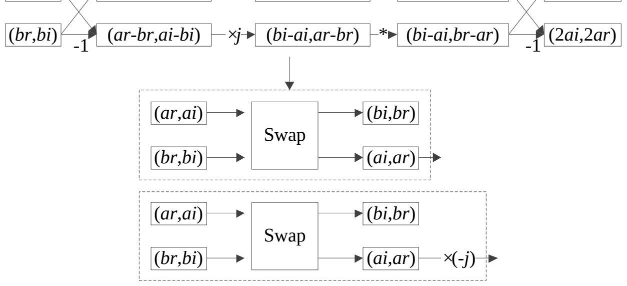

Fig.2. Stages optimization of discrete HT SFG

is of DIT decomposition.

In the middle of discrete HT SFG, three stages without twiddle factor multiplications are combined into one stage, so the stages of N-point discrete HT are changed from $2\log_2 N + 1$ to $2\log_2 N - 1$. The optimization decreases two stages iteration in the discrete HT SFG.

3.2 Optimization of the discrete HT SFG

The discrete HT SFG described above appears regularity and has less complex multipliers required. Thus, it is suited for hardware implementation. The optimization of the discrete HT SFG can decrease the computation latency significantly. Discrete HT in the frequency domain is expressed in equations (6) and (7). It is evident that the discrete HT can be calculated easily in the frequency domain as multiplications with $\pm j$, and these multiplications can be replaced with conjugating and swapping operations. Moreover, the last FFT stage and the first IFFT stage have multiplications with $\pm j$ also. Discrete HT multiplication in the frequency domain can be combined with two adjacent FFT/IFFT stages. This method is shown in Fig.2. Assume that the inputs of one butterfly in the last FFT stage are (*ar*, *ai*) and (*br*, *bi*), so the FFT butterfly result is (*ar*+*br*, *ai*+*bi*) and (*ar*-*br*, *ai*-*bi*). After complex multiplications in the frequency domain, the result is (*ai*+*bi*, -*ar*-*br*) and (*bi*-*ai*, *ar*-*br*). One conjugation is performed after complex multiplications, so the numbers are changed to (*ai*+*bi*, *ar*+*br*) and (*bi*-*ai*, *br*-*ar*). Then the butterly in the first IFFT stage treats (*ai*+*bi*, *ar*+*br*) and (*bi*-*ai*, *br*-*ar*) as the inputs and this IFFT butterfly results are (2*bi*, 2*br*) and (2*ai*, 2*ar*). The results can write as (*bi*, *br*) and (*ai*, *ar*) also. A series of complicated operations on (*ar*, *ai*) and (*br*, *bi*) are optimized to swap operations shown in Fig.2. We call this optimization as "Swap".

3.3 Twiddle Factors Symmetry

Twiddle factors have a symmetric property. In the second quadrant, twiddle factor multiplications with a complex number can be written as:

$$W_N^k(a+jb) = W_N^{k-(N/4)}(b-ja),\ N/4 \le k < N/2 \quad (13)$$

Twiddle factors is located in the first and the second quadrant. Given the equation (13), twiddle factors in the second quadrant can be obtained by a combination of twiddle factors in the first quadrant. In other words, arbitrary twiddle factors used in discrete HT can utilize this operation type to derive the wanted value, thus can significantly shorten the size of ROM used to store the twiddle factors. Based on the symmetric property, the ROM size for twiddle factors will be reduced half.

3.4 The Proposed discrete HT Architecture

A 16-point pipelined discrete HT processor is shown in Fig.3. The proposed architecture is composed of five different types of processing elements (PEs), delay-line (DL) buffers (as shown by a rectangle with a number inside), and some other units. The proposed architecture is also suited for arbitrary point sizes in powers-of-2 by adding some PE3s to FFT and IFFT. Here, PE0 is a module without twiddle factor multiplication, and it is a sub-module of PE1, PE2 and PE3. These three types of PEs are modules that contain twiddle factor multiplications, and they are divided into two parts internal. One part is complex additions and subtractions shown by a rectangle with a string “PE0” inside. Another part is twiddle factor multipliers shown by rectangles with letter “M” followed by a number inside. The star symbol in the figure means a conjugating operation which is easy to implement by taking the 2’s complement of the imaginary part of a complex value. The divided-by-16 module can be substituted with a shifter. The detailed functions and structures of five types of PEs and three types of twiddle factor multipliers in Fig.4 are described in the following subsections.

3.5 The Pipelined butterfly PE architecture

Based on the radix-2 FFT algorithm, five types of PEs (PE0, PE1, PE2, PE3 and PE4) used in our design are illustrated from Fig.4 to Fig.8. The functions of these five PE types correspond to each of the butterfly stages as shown in Fig.1.

The pipelined structure of PE0 stage is shown in Fig.4. PE0 is used to perform the complex additions of the butterfly, and serves as the sub-modules of other PEs. (r_{in}, i_{in}) is the complex input data, (R_{out}, I_{out}) is the complex output data. (R0, I0) and (R1, I1) are one pair complex output data in the figure. Complex input data will store to DL buffers firstly until DL buffers are full. By this way, the input data has been broken into two parallel data stream flowing forward, with same length and correct “distance” between data elements entering the adders by proper delays.

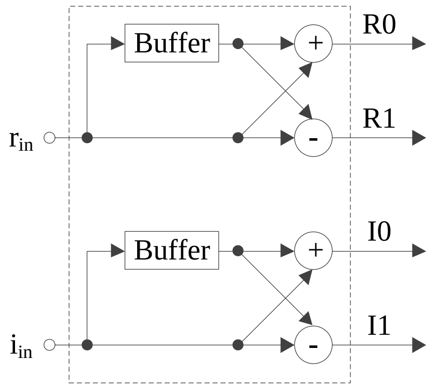

Fig.4. The architecture of PE0

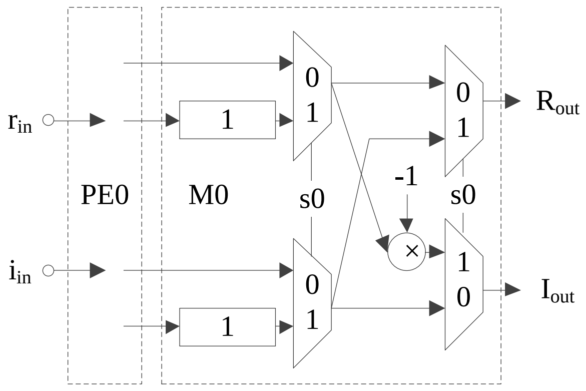

Fig.5. The architecture of PE1

As for the PE1 stage shown in Fig.5, M0 is a sub-module to compute multiplication by *–j*. Data (R0, I0) from PE0 is send to M0 firstly. At the same time, data (R1, I1) is stored in buffers in M0. The length of buffers in M0 is 1. If signal “s0” is 0, data from PE0 will be sent to (R_{out}, I_{out}) directly. If signal

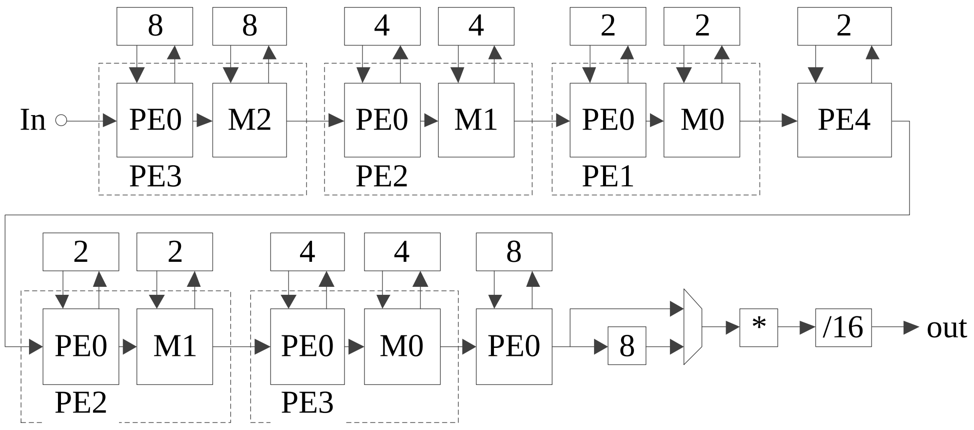

Fig.3. The architecture of proposed discrete HT processor

"s0" is 1, data from DL buffers will be multiplied by –*j*, and then sent to (R_{out}, I_{out}). DL buffers in Fig.5 are used to combine two parallel complex data (R0, I0) and (R1, I1) into one successive serial data.

The pipelined structure of the PE2 is shown in Fig.6. "M1" is a sub-module used to compute the twiddle factor multiplications with $W_N^{N/8}, W_N^{3N/8}$ or -*j*. Since $W_N^{3N/8} = -jW_N^{N/8}$, the multiplication by $W_N^{N/8}$ and then by –*j* can substitute for the multiplication with $W_N^{3N/8}$. Hence, the structure of PE2 utilizes this kind of cascaded calculation and some multiplexers to realize all the necessary calculations in the PE2 stage. This method saves one complex multiplier to

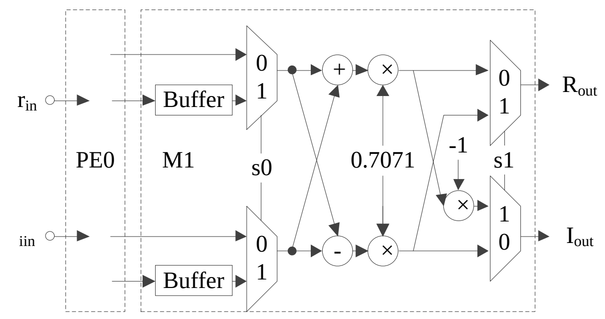

Fig.6. The architecture of PE2

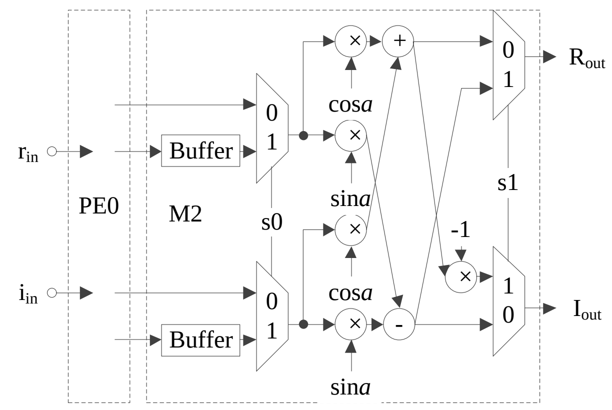

Fig.7. The architecture of PE3

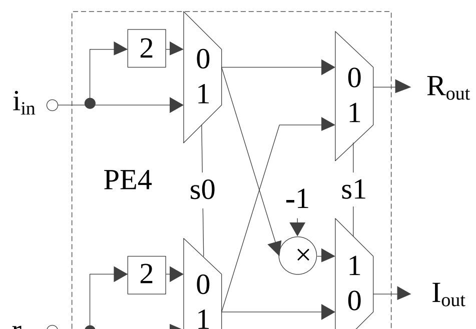

Fig.8. The architecture of PE4

TABLE 1
TWIDDLE FACTORS SELECTION

s0	s1	Twiddle factors
0	0	1
0	1	-*j*
1	0	$W_N^{N/8}$
1	1	$W_N^{3N/8}$

forms a low-cost hardware for computing $W_N^{3N/8}$. In the figure, signal "s0" is used to select data from (R0, I0) or DL buffers. Signal "s0" is also used to enable or disable adders and multipliers in "M1". Signal "s0" and "s1" works together to control the twiddle factors to be multiplied shown in Table 1. If "s1" and "s2" are both zero, meaning the twiddle factor is 1, no multiplication need to perform. If "s1" is zero and "s2" is one, then "M1" performs the complex multiplications by -*j*. If "s1" is one and "s2" is zero, then "M1" performs the complex multiplications by $W_N^{N/8}$. If "s1" and "s2" are both one, then "M1" performs the multiplications by twiddle factor $W_N^{3N/8}$.

The pipelined structure of the PE3 is shown in Fig.7. "M2" is a sub-module used to compute twiddle factors multiplications which is composed of four multipliers and two adders. One ROM is required for storing twiddle factors. PE3 is a fully functional DIF butterfly which support arbitrary twiddle factor multiplications.

PE4 is shown in Fig.8. PE4 has no multipliers and adders. The function of PE is that it swaps the real part and the imaginary part of complex data, and reverses the order of two successive complex data meanwhile. These operations are controlled by "s0". Signal "s1" is used to control whether to do multiplication by –*j*.

3.6 General multiplier

Multipliers are used in PE3 for the computation of complex multiplication by twiddle factors. Based on equation 13, the circuit structure of complex multiplier shown in Fig.9, also adopts a cascaded scheme to achieve low-cost hardware. Here, (I0, I1) are the input signals, and (Q0, Q1) are the output signals the same as (R_{out}, I_{out}) in PE3. (cos*a*, sin*a*) are twiddle factors read from ROM.

Some architectures are proposed in [22-26] Serial, booth and carry save are some general architectures in multiplier design. Serial multiplier has a high clock rate, but it has a large circuit area

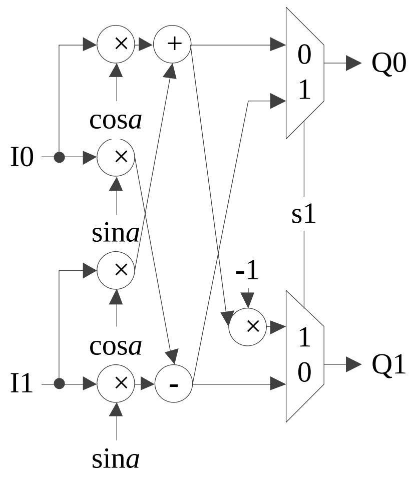

Fig.9. The architecture of the complex multiplier.

and long latency for computation. 4-bit booth multiplier saves half of the clock cycles compared with serial multiplier, but the latency is not the minimal. The array multiplier has a compacted structure and a very efficient layout. But it is hard to determine the propagation delay straightforward due to array organization. The carry save architecture is chosen for designing the 16x16 bit multiplier because it has a moderate latency with comparatively small area. The carry bits are passed diagonally downwards in the carry save multiplier. Partial products are made by anding the inputs together and passing them to the appropriate adder.

The number of adders (HA and FA) in each stage is equal to the mantissa's length minus one. For example, a 16x4 carry save multiplier is shown in Fig.10 and it has four stages: The first stage consists of three HAs. The second and the third stages consist of three FAs respectively. The last stage consists of one HA and two FAs. The critical path of the processor is in the multiplier. The latency of the proposed pipelined multiplier is L_M=4.

3.7 Adders

The design and simulation of various adder structures are depicted in [27-36]. The carry-lookahead structure has the fast speed, but it is only useful for small input words width. The carry-select structure is chosen to implement the 16-bit adder in PEs. The carry-select adder anticipates both possible values of the carry input and evaluate the result for both possibilities in advance. Once the real value of the incoming carry is known, the correct result is easily selected with a simple multiplexer. The latency of the carry-select adder is L_A=1.

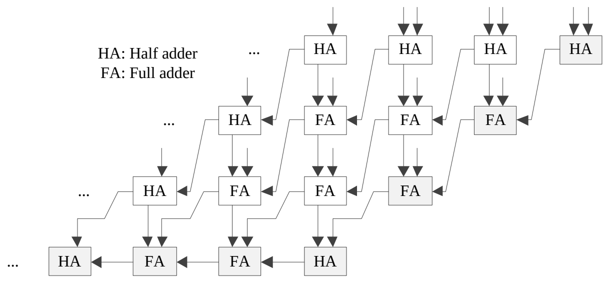

Fig.10. the architecture of 16x4 bit carry save multiplier

3.8 Constant multipliers

Constant multipliers have higher computed speed and less chip area cost than general multipliers. The constant multiplication by $\sqrt{2}/2$ can be implemented by a special bit parallel multiplier instead of general word length multipliers. The binary representation of $\sqrt{2}/2$ is:

$$\sqrt{2}/2 = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-14} \quad (21)$$

Eqn.21 includes five additions and six shift operations, and a straightforward implementation for the above equation introduces a poor precision due to the truncation error [17], and spends more hardware cost. To improve the precision, eqn.19 can be rewritten as:

$$\sqrt{2}/2 = (2^{-1} + 2^{-3})(1+2^{-3}) + 2^{-3}(2^{-5}+2^{-11}) \quad (22)$$

According to eqn.22, the circuit structure of the constant multiplier is illustrated in Fig.11. The circuit uses four adders and five barrel shifters. The constant multiplier adopts pipeline technology for high performance, and its latency is three clock cycles. The realization of complex multiplication by $W_N^{N/8}$ and $W_N^{3N/8}$ using a radix-2 butterfly structure with its both outputs multiplied by $\sqrt{2}/2$ is shown in Fig.6. This circuit structure has just been used in the PE2 and its latency is $L_{0.707}$=3.

The multiplication by *-j* can be calculated by a negation and an additions, so its latency is L_{-j}=1.

4 Implementation and Performance Analysis

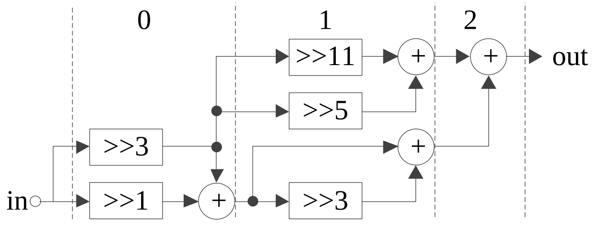

Fig.11. The architecture of the constant multiplier

4.1 Implementation

A pipelined discrete HT processor with point size 1024 is implemented in the paper. The word width of the processor is 16 bits. The processor is simulated on XC5VLX50T-2 FPGA. The placement, route process, and timing analysis of the synthesized designs are accomplished using Xilinx Design Suite 13.2. The design was written by Verilog HDL and doesn't contain Xilinx DSP48Es.

The latency of the 1024-point discrete HT processor is the sum of all stages latency of PEs and the point size. The latency of each stage is the clock cycles of cascaded arithmetic units cost and the depth of buffers. The latency of all stages is listed in table 2. So the processor latency of the 1024-point FFT and discrete HT are

$$L_{FFT} = 1024 + \sum_{i=0}^{9} L_i = 3119 \qquad (23)$$

$$L_{HT} = 1024 + \sum_{i=0}^{18} L_i = 5212 \qquad (24)$$

4.2 Comparison

In Table 3, a number of DSP processors are compared with the proposed one. The hardware in [37] has a low clock rate. This architecture adopts radix-4 FFT algorithm and has poor memory utilization. The architecture includes four memories, and their depth equals to the discrete HT point size. The FFT processor proposed in [38] adopts ASIC implementation with 0.18um and 1.8V voltage. This FFT processor has a slow latency because the PE is of radix-2^4 algorithm. In [39] the PE is not working in pipelined mode, so the latency is longer than our work. In our work, the 1024-point pipelined discrete HT includes 19 stages after the optimization of SFG. In Table 3, our work has the minimal latency for computing discrete HT.

TABLE 2
THE LATENCY OF EACH STAGE IN 1024-POINT DISCRETE HT

Stages	Latency
0, 18	L_0=L_{18}=2*512+2L_A+L_M=1030
1, 17	L_1=L_{17}=2*256+2L_A+L_M=518
2, 16	L_2=L_{16}=2*128+2L_A+L_M=262
3, 15	L_3=L_{15}=2*64+2L_A+L_M=134
4, 14	L_4=L_{14}=2*32+2L_A+L_M=70
5, 13	L_5=L_{13}=2*16+2L_A+L_M=38
6, 12	L_6=L_{12}=2*8+2L_A+L_M=22
7, 11	L_7=L_{11}=2*4+2L_A+$L_{0.707}$=13
8, 10	L_8=L_{10}=2*2+L_A+L_{-j}=6
9	L_9=2+L_{-j}=2

5 Conclusion

A high performance pipelined discrete HT processor is presented in this paper. The processor adopts FFT algorithm to compute discrete HT. Some stages in the discrete HT SFG have no multiplications. They are combined into one stage to decrease the computation latency. The discrete HT processor is composed of four types pipelined PE. Data being processed is of fixed point mode with 16-bit word width. Pipelined adders and multipliers are designed to enhance the performance of the discrete HT processor. The proposed discrete HT processor is written in Verilog HDL, so it is easy for ASIC implementation. The performance analysis with some previous paper approaches show that the proposed discrete HT processor has the shortest clock latency in discrete HT computation with same samples.

References:

[1] S.C. Kak, The discrete Hilbert transform, Proc. IEEE, Vol.58, 1970, pp.585-586.

[2] V. Cizek, Discrete Hilbert transform, IEEE Trans. Audio and electroacoustics, Vol.18, No.4, 1970, pp.340-343.

[3] A.V. Oppenheim, R.W. Schafer, Discrete time

TABLE 3
PERFORMANCE COMPARISON OF DIFFERENT DISCRETE HT PROCESSORS

Works	Point	Clock rate *MHz*	FFT Latency	FFT time (*us*)	Discrete HT latency	Discrete HT time (*us*)	REG	LUT	DSP	FPGA
[37]	1024	89	-	14.1	-	28.2	-	-	-	EP200K400E
[38]	1024	110		92.7						
[39] Xilinx	1024	366	7364	20.12	14728	37.44	1253	1114	3	XC5VLX50T-2
Our work	1024	347	3119	8.988	5212	15.02	5268	6486	0	XC5VLX50T-2

signal processing, Prentice Hall, 1989, pp.775-810.

[4] R.G. Lyons, Understanding digital signal processing, Prentice Hall, 2005, pp.362-364

[5] J.W. Cooley, J.W. Tukey, An Algorithm for the Machine Calculation of Complexes Fourier Series, Math. Computation, Vol.19, No.90, 1965, pp.297-301.

[6] S. Winograd, On Computing the DFT, Math. Computation, Vol.32, No.141, 1986, pp.175-199.

[7] D. Kolba, T. Parks, A Prime Factor Algorithm using High-speed Convolution, IEEE Trans. Acoust. Speech Signal Process, Vol.25, No.4, 1977, pp.281-294.

[8] S.C. Pei, S.B Jaw. Computation of the discrete Hilbert transform through fast Hartley transform, IEEE Trans. Circuits and Systems, Vol.36, No.9, 1989, pp.1251-1252.

[9] B. Kumar, S.C Dutta Roy, Design of efficient FIR digital differentiators and Hilbert transformers for midband frequency ranges, International Journal of Circuit Theory and Application, Vol.17, No.4, 1989, pp.483-488.

[10] S.K. Padala, K.M.M Prabhu, Systolic arrays for the discrete Hilbert transform, Proc of IEEE CDS, Vol.144, No.5, 1997, pp.259-264.

[11] S. He and M. Torkelson, “Designing Pipeline FFT Processor for OFDM (de)Modulation,” in Proc. URSI Int. Symp. Signals, Systems, and Electronics, Vol.29, 1998, pp.257-262.

[12] H.L. Groginsky and G.A. Works, “A pipeline fast Fourier transform,” IEEE Transactions on Computers, Vol.19, No.11, 1970, pp.1015-1019.

[13] H. Shousheng, Design and implementation of a 1024-point pipeline FFT processor, Custom Integrated Circuits Conference, 1998, pp.131-134.

[14] K. Maharatna, E. Grass, U. Jagdhold, A 64-Point fourier transform chip for high-speed wireless LAN application using OFDM, IEEE Journal of Solid-State Circuits, Vol.39, no.3, 2004, pp.484-493.

[15] Y.T. Lin, P.Y. Tsai and T.D. Chiueh, “Low-power variable-length fastFourier transform processor,” IEEE Proc. Comput. Digit. Tech., Vol.152, No.4, 2005, pp.499-506.

[16] S. Minhyeok, L. Hanho, A High-Speed Four-Parallel Radix-24 FFT/IFFT Processor for UWB Applications, in Proc. IEEE Int. Symp. Circuits and Systems, 2008, pp. 960-963.

[17] B.M. Bass, A low-power, high performance 1024-point FFT processor, IEEE Journal of Solid-State Circuits, Vol.34, No.3, 1999, pp.380-387.

[18] M. Hasan, T.Arslan, J.S. Thompson, A novel coefficient ordering based low power pipelined radix-4 FFT processor for wireless LAN applications, IEEE Transactions on Consumer Electronics, Vol.49, No.1, 2003, pp.128-134.

[19] S. He, M. Torkelson, Designing Pipeline FFT Processor for OFDM (de)Modulation, Proc. IEEE URSI Int. Signals, Systems, and Electronics, Vol. 29, 1998, pp.257-262.

[20] E.H. Wold, A.M. Despain, Pipeline and parallel-pipeline FFT processors for VLSI implementation, IEEE Trans. Computer, Vol.33, No.5, 1984, pp.414-426.

[21] Y. Chu, L. Yiting, Y. Maohsu, H. Paoann, et al, A Low-Power 64-point Pipeline FFT/IFFT Processor, IEEE Transactions on Consumer Electronics, Vol.57, No.1, 2011, pp.40-45.

[22] W.C. Yeh, C.W. Jen, High Speed Booth Encoded Parallel Multiplier Design, IEEE transactions on computers, Vol.49, No.7, 2000, pp.692-701.

[23] Z. Haung, M.D. Ercegovac, High performance Low Power left to right array multiplier design, IEEE Trans. Computer, Vol.54, No.3, 2005, pp.272-283.

[24] N.H.E. Weste, D. Harris, A. Banerjee, CMOS VLSI Design A circuits and Systems Perspective, Third edition, Pearson Education, pp.347-349.

[25] A.D. Booth, A Signed Binary Multiplication Technique, Quartely Journal of Mechanics and Applied Mathematics. Vol.4, No.2, pp.236-240, 1951.

[26] O.L. MacSorley, High Speed Arithmetic in Binary Computers, Proc of the IRE, Vol.49, no.1, 1961, pp. 67-97.

[27] T.Y. Chang, M.J. Hsiao, Carry-select adder using single ripple-carry adder, Electronics Letters, vol.34, No. 22, 1998, pp.2101-2103.

[28] J.M. Rabaey, Digital Integrated Circuits: A Design perspective. New Jersey, Prentice-Hall, 1996.

[29] H. Morinaka, H. Makino, Y. Nakase, et al, A 64 bit Carry Look-ahead CMOS adder using Modified Carry Select. Custom Integrated Circuit Conference, 1995, pp.1985-1993.

[30] K. Youngjoon, K.L. Sup, A low power carry select adder with reduced area, Proc of ISCAS 2001. Vol.4, 2001, pp218-221.

[31] R.P.P. Singh, P. Kumar, B. Singh, Performance Analysis of Fast Adders Using VHDL, Proc of ARTcom, 2009, pp.189-193.

[32] P. Prashanth, P. Swamy, Architecture of adders based on speed, area and power dissipation, Proc of WICT 2011, 2011, pp.240-244.
[33] P. Gurjar, R. Solanki, P. Kansliwal, et al, VLSI implementation of adders for high speed ALU, Proc of INDCOM 2011, 2011, pp.1-6.
[34] S. Liyuan, J. Wenming, T. Shuai, et al, A Parallel Feedback Carry Adder Based on Half Adder, Proc of ICIECS 2010, 2010, pp.1-4.
[35] H. Seok-Won, K. Moon-Gyung, L. Yong-Surk, Study of optimized adder selection, Proc of ICASIC 2003, 2003, Vol.2, pp.1265-1268.
[36] C. Pinghua, Z. Juan, X. Guobo, et al, An improved 32-bit carry-lookahead adder with Conditional Carry-Selection, Proc of ICCSE 2009, 2009, pp.1911-1913.
[37] X. Yingke, F. Bo, Design and Implementation of High Throughput FFT Processor, Journal of Computer Research and Development, Vol.41, No.6, 2004, pp.1022-1029 (in Chinese).
[38] L. Qingwang, W. Xin'an, N. Jiuchong, A Low-power Variable-length FFT Processor Base on Radix-24 Algorithm, Proc of PRIMEASIA 2009, 2009, pp.129-132.
[39] Xilinx, Inc. LogiCORE IP Fast Fourier Transform v7.1 [EB/OL], 2011-3-1. http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf

Wang Xu, born in 1980. Received the M.A's. degrees in microelectronics from the Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China, in 2007. Since 2008, he has been a PhD candidate in microelectronics. His main research interests include image processing and embedded DSP processor design.

Zhang Yan, born in 1969. He has been professor of the Shenzhen Graduate School, Harbin Institute of Technology since 2002. His main research interests are application specific instruction set processor design, including medical image processing chips and wireless communication baseband chip.

Ding shunying, born in 1988, she is a BSc candidate in microelectronics in the Shenzhen Graduate School, Harbin Institute of Technology. Her main research interest is image processing and FFT acceleration by Intel SSE and NVidia CUDA.