A Configurable Floating-Point Discrete Hilbert Transform Processor for Accelerating the Calculation of Filter in Katsevich Formula

WANG XU, ZHANG YAN, WANG FEI 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: - Katsevich formula is currently a hot topic for cone-beam computed tomography (CBCT). The filter in the formula can be computed by a regular discrete Hilbert transform (DHT). A configurable single precision floating-point (SPFP) DHT processor is proposed for accelerating the calculation of filter in Katsevich formula. The configurable processor is of memory based architecture with one pipelined butterfly processing engine (PE) and supports variable point sizes from 8 to 1024. The DHT processor is controlled by the address generator. According to the point size, the address generator yields one memory address pair per clock cycle to keep the processor accessing memories successively. The DHT is calculated easily via complex multiplications in the frequency domain. Two fast Fourier transforms (FFT) are required in the entire process. The radix-2 FFT algorithm with decimation-in-frequency (DIF) decomposition is utilized in the design to construct an efficiently signal flow graph (SFG) for DHT calculation. Arithmetic calculations, in the last FFT iteration, complex multiplications and the first IFFT iteration are replaced with conjugation and swapping operations, so two iterations are saved in the DHT SFG. Data are loaded and unloaded simultaneously after one frame data calculation is completed. The symmetric property of twiddle factors is utilized to decrease half size of the read-only memory (ROM). Truncation is used in the design to reduce data path width. The proposed DHT processor is written in Verilog HDL, so it is easy for ASIC implementation. Compared with previous works, the performance analysis shows that the proposed DHT processor has minimum clock latency.

Key-Words: - discrete Hilbert transform, FFT, floating-point adder, floating-point multiplier, ASIC, VLSI

1 Introduction

Katsevich formula is developed in [1][2][3] for fully 3-D CBCT with a helical scanning path. This formula is theoretically exact before discretization, and it may be implemented via a filtered-back projection type algorithm. Therefore, it has the potential for both high accuracy and fast implementation. The filter step is the bottleneck of Katsevich formula because it can be seemed as a regular 1-D DHT.

The DHT was developed by Kak, Cizek, and Oppenheim [4][5][6] for applying digital signal processing (DSP) techniques to analytic signal, minimum phase sequence etc. DHT 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 DHT [7].

The most widely used method for computing the DHT is through the use of the FFT. Since the early paper by Cooley and Tukey [8], a large number of FFT algorithms have been developed such as radix-2 algorithms, Winograd algorithm (WFTA) [9], prime factor algorithms (FPA) [10], and fast Hartley transform (FHT) [11]. These methods use different transforms to compute the DHT, their basic method of computing the DHT is that they all use the transform domain for computing the DHT. There are other methods, such as the filter method [12] and the systolic arrays [13]. This method comes directly from the DHT definition. This method is the direct implementation of the convolution operation on the input sequence with the impulse response of the Hilbert transformer [6]. 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.

For hardware implementation, architectures of DHT processor based on FFT algorithms can be generally grouped into pipelined and memory based architecture styles. Various FFT processors have been proposed in [14]-[23]. The DHT processors have two popular design styles. One is single-path delay feedback (SDF) pipelined architecture [14] [15], and the other is multi-path delay commutator (MDC) pipelined architecture [16]. Pipelined architectures are good at memory requirement, as well as advantage to low power design, but they are usually used to design a fixed point size DHT processor. Memory based architectures are widely used to design configurable DHT 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.

A configurable SPFP DHT processor is proposed to accelerate the calculation of filter in Katsevich algorithm. The processor is of memory based architecture with one pipelined butterfly PE and supports variable point sizes from 8 to 1024. The DHT processor is controlled by the address generator. According to the point size, the address generator yields one memory address pair per clock cycle to keep the processor accessing memories successively. The DHT is calculated easily via complex multiplications in the frequency domain. Two FFTs are required in the entire process. The radix-2 FFT algorithm with DIF decomposition is utilized in the design to construct an efficiently SFG for DHT calculation. Arithmetic calculations, in the last FFT iteration, complex multiplications and the first IFFT iteration are replaced with conjugation and swapping operations, so two iterations are saved in the DHT SFG. Data are loaded and unloaded simultaneously after one frame data calculation is completed. The symmetric property of twiddle factors is utilized to decrease half size of the ROM. Truncation is used in the design to reduce data path width. The proposed DHT processor is written in Verilog HDL, so it is easy for ASIC implementation. Compared with previous works, the performance analysis shows that the proposed DHT processor has minimum clock latency.

The rest of this paper is organized as follows. In the next section we review the DHT definition, and then discuss its related computational methods and algorithms. IEEE standard for SPFP arithmetic is also introduced briefly. In section III a novel DHT SFG and the architecture of the proposed DHT processor is illustrated. Then the pipelined butterfly PE, SPFP multipliers, SPFP adders, and memory address generator are presented in detail in this section. Performance evaluation and comparison of various DHT architectures is presented in section IV. Finally, concluding remarks are given in Section V.

2 DHT Algorithms and IEEE 754 Standard

This section gives a brief review on definitions and computational methods of DHT based on FFT/IFFT. IEEE 754 standard for SPFP is also discussed here.

2.1 The Discrete Hilbert Transform

The Hilbert transform of signal $x(t)$ is defined as

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

where $\hat{x}(t)$ is the Hilbert transform result [6].

The Hilbert transform 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 DHT 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 DHT given by

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

The DHT can be computed via FFT as shown below

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

Where $X(k)$ = FFT($x(n)$) and

$$-j\,\mathrm{sgn}(k) = \begin{cases} -j & k = 1,2,\ldots,N/2-1 \\ 0 & k = 0,\ N/2 \\ +j & k = N/2+1,\ldots,N-1 \end{cases} \quad (7)$$

It is evident that the DHT 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 $X(k)$ of a sequence $x(n)$ in the time domain. The N-point DFT and Inverse DFT (IDFT) are defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk},\ k = 0,1,\ldots,N-1 \qquad (8)$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-nk},\ n = 0,1,\ldots,N-1 \qquad (9)$$

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

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

(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 calculate in (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 [8] in 1965. FFT is an efficient approach for reducing the computational complexity of DFT. FFT has many flexible algorithms. Generally, FFT treats input sequence by using decimation-in-frequency (DIF) or decimation-in-time (DIT) decomposition to build a regular SFG. Radix-2 DIF FFT is chosen in this paper because we analyze the input sequence in natural ordering in most cases.

2.3 IEEE Standard for Binary Single Precision Floating-Point Arithmetic

An IEEE binary SPFP number [24] is a 32-bit word with 1 bit for sign "*s*", 8 bits for exponent "*e*" with 127 bias and 23 bits for mantissa "*m*" with true binary notation. A normal SPFP number is represented in

$$n = (-1)^s \times 2^e \times (1+m), \quad (0 < e < 255) \qquad (11)$$

Denormalized SPFP numbers are too small. In most practical calculation they do not contribute to the final result, so we treat denormalized numbers as zero, with a sign "*s*" taken from them. Some special SPFP numbers are depicted as follows.

1. $e = 255, m \neq 0$, not a number (NAN).
2. $e = 255, m = 0$, infinity depending on "*s*".

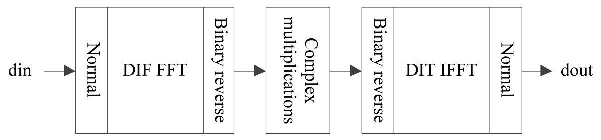


Fig.1. 16-point DHT SFG by a DIF FFT and a DIT IFFT

3. $e = 0, m = 0$, zero depending on "*s*".
4. $e = 0, m \neq 0$, denormalized operands.

SPFP arithmetic operation has some rounding modes. Truncation and convergent rounding are two mostly used modes for keep the result precision. In hardware implementation, truncation is the easiest method for hardware implementation. The pipelined butterfly PE of the proposed DHT processor use truncation to reduce data path width.

3 The Proposed DHT SFG and associated Architecture

3.1 The Proposed DHT SFG

Bit reversed sorting is a complicated operation in radix-2 FFT algorithm. FFT and IFFT are calculated one time in respective in the DHT computation. A popular DHT SFG is depicted in Fig. 1. In this figure, the input and the output of the complex multiplication module are both bit reversed order, so no bit reversed sorting is used. The main disadvantage is that Fig. 1 uses both DIF and DIT decomposition. DIF PE and DIT PE have different arithmetic operation sequence, so the hardware implementation using Fig. 1 may wastes hardware resource.

As an example, a novel 16-point DHT SFG is proposed in Fig. 2. FFT and IFFT in this figure are both using radix-2 FFT with DIF decomposition. Input and output of the complex multiplication module are both bit reversed order. FFT and IFFT in Fig. 2 share one pipelined DIF butterfly PE while avoiding bit reversed sorting in DHT computation. Another advantage in Fig. 2 is that the input and output sequence are both in natural order.

3.2 Optimization of Complex Multiplications in the Frequency Domain

Complex multiplication in the frequency domain is expressed in (6) and (7). It is evident that the DHT can be calculated easily in the frequency domain as multiplications with +*j*, -*j* or 0. Actually, complex multiplication is implemented by some easily circuits without multipliers in this paper.

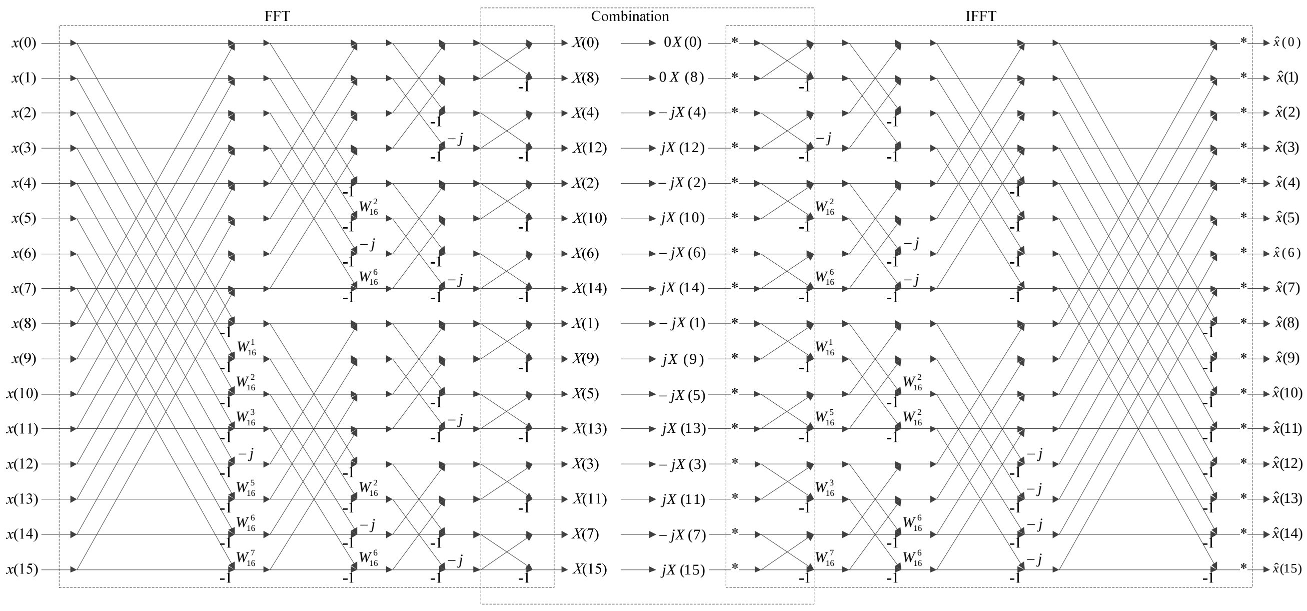


Fig.2. The 16-point DHT SFG by two radix-2 DIF FFTs

Decrease iterations in Fig. 2 can decrease the DHT computation latency significantly. In the figure, the last FFT iteration and complex multiplications is calculated without multiplications. We merge the two iterations into the first IFFT iteration. This skill is shown in Fig. 3. Assume that the inputs of one butterfly in the last FFT iteration are (*ar*, *ai*) and (*br*, *bi*), so the butterfly result is (*ar*+*br*, *ai*+*bi*) and (*ar*-*br*, *ai*-*bi*). Complex multiplications in the frequency domain are replaced with swapping and conjugation operations. The result of this step is (*ai*+*bi*, -*ar*-*br*) and (*bi*-*ai*, *ar*-*br*). Two numbers are conjugated to (*ai*+*bi*, *ar*+*br*) and (*bi*-*ai*, *br*-*ar*). Then the butterly in the first IFFT iteration treats (*ai*+*bi*, *ar*+*br*) and (*bi*-*ai*, *br*-*ar*) as the inputs and this 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 changing (*ar*, *ai*) and (*br*, *bi*) to (*bi*, *br*) and (*ai*, *ar*) are optimized to swapping operations easily. The optimization of complex multiplications in the frequency domain decreases two iterations in the DHT SFG shown in Fig. 2. We call this optimization as swapping operation. If the DHT point size is 2^n, in other words the DHT SFG has *n* iterations. The proposed processor performs *n*-2 iterations after the DHT SFG optimization.

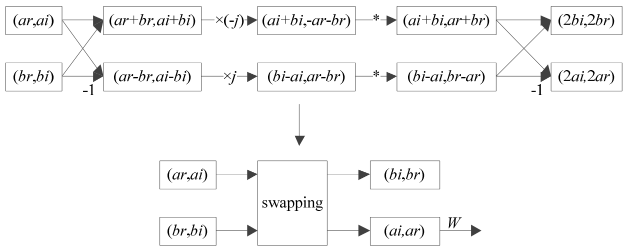


Fig.3. Optimization of complex multiplication in the frequency domain

The last IFFT iteration doesn't contain any complex multiplication. The DHT results can be stored in memory using bypass circuits.

3.3 The Proposed DHT Architecture

The architecture of the proposed DHT processor is shown in Fig. 4. It mainly employs one single port ROM, two random access memories (RAM), three crossbar switches, one pipelined butterfly PE and one memory address controller. Two RAMs are used to read and write data and ROM is used for twiddle factors accessing. The width of two RAMs and ROM are both 64-bit so as to access the real and image part of complex data simultaneously. The RAM is dual-port, one port is used to read data and

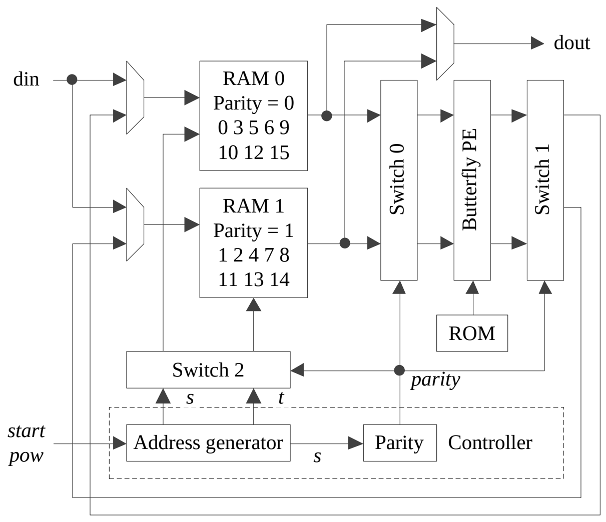


Fig.4. The architecture of proposed DHT processor

another is used to write data. This method avoids memory access conflict when the pipelined butterfly PE working in one stage iteration of the DHT SFG. The controller includes an address generator and a parity module. The parity of element *X* is defined as zero if the number of ones in the binary representation of *X* is even and one otherwise. "*din*" is the data load port and "*dout*" is the data unload port. The strobe signal "*start*" notices the processor to work after memory initiation. The point size of DHT computation is "2^*pow*" determined by the signal "*pow*". The address generator makes an address pair (s,t) per clock cycle. The signal "*parity*" generated by the parity module controls memories accessing, three crossbar switches and the multiplexer connected to unload port. The memory initial rule is that the data is loaded to RAM0 with "*parity*" is zero and to RAM1 with "*parity*" is one. The signal "*parity*" also means swapping data before and after the butterfly PE working. Because the data read from RAM0 and RAM1 may be have a reversed order comparing to the DHT SFG in Fig. 2. Data read from memory is delivered to the butterfly PE every clock cycle to keep the pipeline working in one stage iteration in the DHT SFG. After computation finished, result in two memories is unloaded in the "*parity*" control.

3.4 Architecture of the Memory Address Controller

The DHT processor supports variable point sizes. The internal memory address controller configures the processor according to the point size *N*. The controller mainly includes a memory address generator and a parity module. The memory address generator makes $N/2$ different address pairs in one stage iteration according to the point size *N*. The parity module controls all crossbar switches.

Cohen [25] proposed a simple way to control memory address. Let an address pair (s,t) represent a DIF butterfly PE operation.

$$X'(s) = (X(s) + X(t))W \qquad (12)$$

$$X'(t) = (X(s) - X(t))W \qquad (13)$$

The value of the address pair in the *i*th butterfly in the *j*th iteration is defined as

$$s = rotate_n(i, j) \qquad (14)$$

$$t = rotate_n(i + N/2, j) \qquad (15)$$

$$n = \log_2 N, i = 0,1,\cdots(N/2-1), j = 0,1,\cdots(n-1)$$

Where *N* is the point size and $rotate_n(i,j)$ is the value of *i* cycling rotated right, by *j* bits, within *n* bits, e.g., $rotate_4(1,1) = 8$ and $rotate_4(9,2) = 6$. The

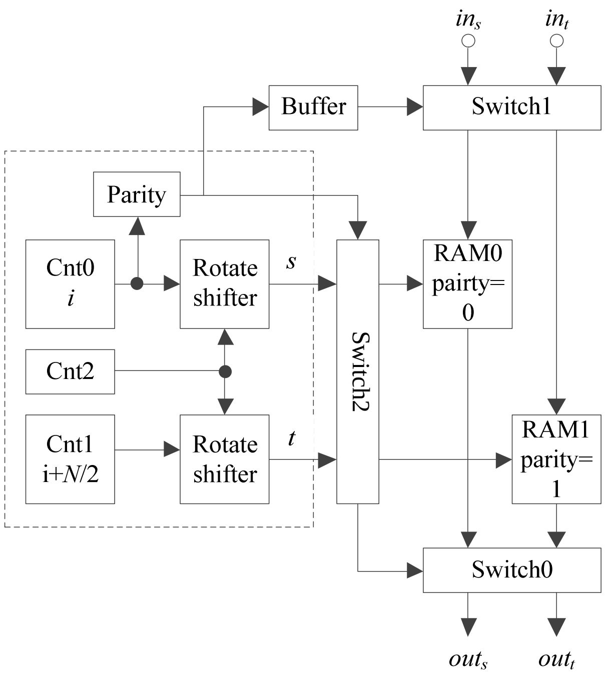


Fig.5. The architecture of memory address generator and parity module

parity of *i* is defined as zero if the number of ones in the binary representation of *i* is even and one otherwise. Pease [26] proved that *s* and *t* in the address pair have different parity.

The architecture of memory controller is shown in Fig. 5. The controller mainly includes three counters, two cycling shifters and one parity module. Continuous numbers from 0 to $N/2-1$ are generated by the counter "Cnt0". Continuous numbers from $N/2-1$ to $N-1$ are generated by the counter "Cnt1". Two numbers generated by "Cnt0" and "Cnt1" are cycling rotated right according to the value of

TABLE 1
16-POINT DHT MEMORY ADDRESS PAIRS

Iterations	Memory address pairs (s,t)
FFT 0	(0, 8), (1, 9), (2, 10), (3, 11), (4, 12), (5, 13), (6, 14), (7, 15)
FFT 1	(0, 4), (8, 12), (1, 5), (9, 13), (2, 6), (10, 14), (3, 7), (11, 15)
FFT 2	(0, 2), (4, 6), (8, 10), (12, 14), (1, 3), (5, 7), (9, 11), (13, 15)
FFT 3/ IFFT 0	(0, 1), (2, 3), (4, 5), (6, 7), (8, 9), (10, 11), (12, 13), (14, 15)
IFFT 1	(0, 2), (4, 6), (8, 10), (12, 14), (1, 3), (5, 7), (9, 11), (13, 15)
IFFT 2	(0, 4), (8, 12), (1, 5), (9, 13), (2, 6), (10, 14), (3, 7), (11, 15)
IFFT 3	(0, 8), (1, 9), (2, 10), (3, 11), (4, 12), (5, 13), (6, 14), (7, 15)

counter "Cnt2", then address pair (s,t) is valid. This address pair will be swapped by crossbar switch 2 if the signal "*parity*" is one. After swapping operation, the address pair is sent to RAM0 and RAM1 in respective. Memory address pairs of a 16-point DHT are show in Table 1. As an example, the address generator makes an address pair (8,12) in FFT iteration 1 where the parity of "*s*" is one, so the address pair is swapped to (12,8), and then address 12 is send to RAM0 and address 8 is send to RAM1. The memory address generator uses one clock cycle to set initial values and one clock cycle to make a rotated address pair. The memory address generator costs two clock cycles before the DHT computation.

3.5 Twiddle Factors Symmetry

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

$$W_N^k(a+jb)=W_N^{k-(N/4)}(b-ja),\ N/4\le k<N/2 \quad (16)$$

Twiddle factors is located in the first and the second quadrant. Given the (16), 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 DHT 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.

3.6 The Pipelined butterfly PE architecture

We design a pipelined butterfly PE to improve the performance of the DHT processor. The VLSI architecture of the butterfly PE shown in Fig. 6 is composed of six adders, four multipliers, three XOR gates, eight multiplexers and one crossbar switch. In the architecture, {*ar*, *ai*} and {*br*, *bi*} are data read from RAM0 and RAM1. {*ar*', *ai*'} and {*br*', *bi*'} are data written to RAM0 and RAM1. {*cosa*, *sina*} are data read from twiddle factors ROM. The signal "*dht_zero*" is active means selecting 0 when the address pair is $(0, N/2)$. The signal "*dht_en*" is active in the first IFFT iteration. The signal "*sym_en*" is active when twiddle factors are in the second quadrant. The signal "*wr_conj*" is active when writing data into memories in the last IFFT iteration.

Let an *N*-point DHT be processed by the pipelined butterfly PE. The DHT SFG shown in the Fig. 2 is divided into three steps in Fig. 3. FFT is the first step. In Fig. 5, signal "*dht*_zero", "*dht_en*" and "*wr_conj*" are all disable, so the butterfly PE becomes a radix-2 DIF butterfly. Swapping is the second step, which includes the last FFT iteration, complex multiplication and the first IFFT iteration. In this step, complex multiplication is active, so signal "*dht_en*" is enabled. The signal "*dht_zero*" is active also when the address pair is $(0, N/2)$ in the first IFFT iteration. This signal sets the complex multiplication results to zero. The third step is executing other IFFT iterations. The signal *wr_conj* is active when writing data in the last IFFT iteration.

3.7 The SPFP Adder Architecture

Floating-point addition is a fundamental component of DSP processors and systems. Various algorithms and design approaches [27-33] have been proposed to increase the performance of floating-point adder.

A five stage pipelined SPFP adder is designed in [28] which has an efficient pipeline division. But the clock rate of this adder is low because of its long critical path. The critical path in the processor is located in the SPFP multiplier, the five stage pipelined SPFP adder is not suited. We proposed a seven stage pipelined SPFP adder architecture shown in Fig. 7. Assume that a SPFP addition starts

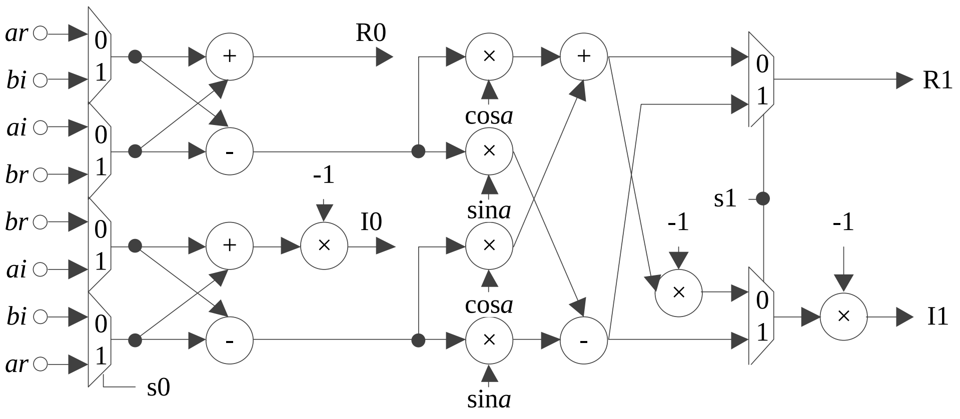


Fig.6. The pipelined butterfly PE architecture

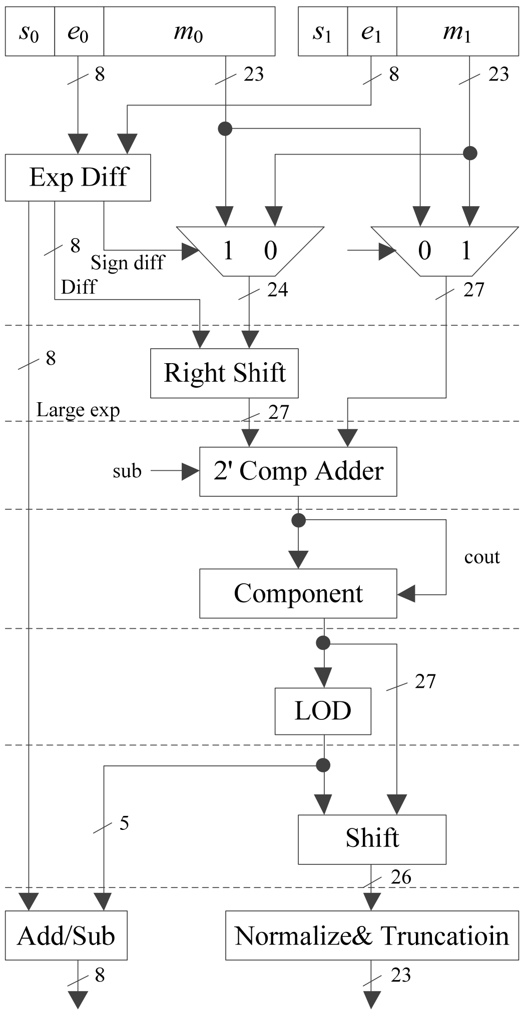


Fig.7. The architecture of SPFP adder

in clock cycle *i*. Then two inputs are loaded from memory and check whether they are denormalized. Denormalized inputs will be replaced with zero. The exponents are subtracted from one another to get the absolute difference and identify the larger exponent. The fraction parts of two inputs are also changed to 2's complement code representation. In cycle $i+1$, the mantissa with smaller exponent is right-shifted to ensure that two inputs have the same exponential. The mantissa with smaller exponent is extended by 3 bits to be used later for rounding. In cycle $i+2$, two mantissas are added using a 2's complement adder. In cycle $i+3$, the sum of two mantissas is changed to true code representation. In cycle $i+4$, detecting the leading number of zeros before the first 1 in the sum, this step is done by the module known as the leading-one detector (LOD). Using this value, the result is left-shifted by a left shifter in cycle $i+5$. Normalization and truncation is finished at cycle $i+6$. The latency of the proposed pipelined SPFP adder is $T_{add} = 7$.

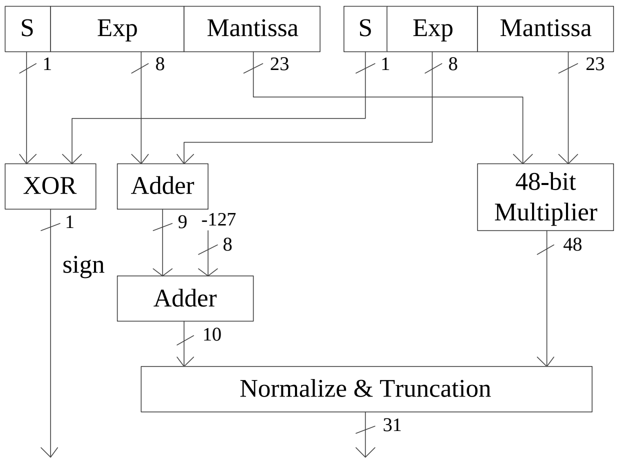


Fig.8. The architecture of SPFP multiplier

3.8 The SPFP Multiplier Architecture

The pipelined architecture of the SPFP multiplier is shown in Fig. 8. The first step, not shown in the figure, is preprocessing. This step includes replacing denormalized operands with zero, then setting the result to zero if one of the two operands is zero or to infinity if one of the two operands is infinity and another is nonzero. Next, we get the exponent result by adding the exponent of the two operands and then subtracting the bias from their sum. The mantissa's product is performed by a fixed point multiplier at the same time. We exclusive or (XOR) the sign of the two operands to get the sign result of the product.

The fixed point multiplier is responsible for multiplying the mantissas and placing the decimal point in the product. Serial, booth and carry save are general multiplier architectures in hardware design. Serial multiplier has a high clock rate, but it has a large circuit area 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 the array organization. Three 24x24 bit multiplier architectures are implemented and their performance comparison is listed in Table

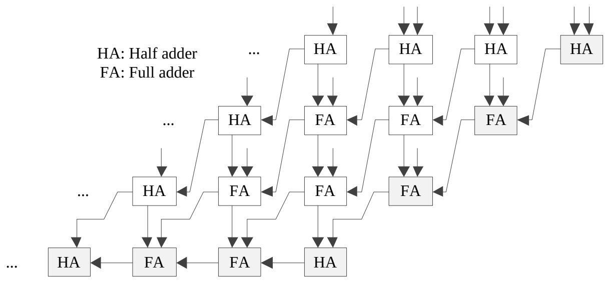


Fig.9. The 24x4 bit carry save multiplier architecture.

TABLE 2
PERFORMANCE COMPARISON OF DIFFERENT MULTIPLIER ARCHITECTURES

Designs	Clock rate (*MHz*)	Latency	LUTs	REGs
Serial	367.834	26	1213	627
4-bit booth	348.372	14	910	524
Carry save	334.316	8	867	476

II. We chose carry save architecture for designing 24x24 bit multiplier because it has a moderate latency with comparatively small area. The carry bits are passed diagonally downwards in 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 24x4 carry save multiplier is shown in Fig. 9 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 latency of the proposed pipelined SPFP multiplier is $T_{mul} = 7$.

4 Implementation and Performance Analysis

4.1 Implementation

The number of clock cycles by itself is not an accurate measure of performance of the system, since the clock rates may vary considerably. So an implementation is carried out to verify the performance of the system.

The circuits are modelled using Verilog HDL. Synthesis uses Xilinx ISE 13.1. The proposed DHT processor is implemented in Xilinx XC5VLX50T with speed grade -2. The design doesn't include any Xilinx DSP slices. The pipeline technique is used in the DHT processor in order to decrease latency, keep the processor working at a higher clock rate, and increase the processor's throughput. The memory address bus width is 16, so the maximum point size the proposed DHT processor supports is 1024.

4.2 Comparison

In Table 3, Some DSP processors are compared with the proposed one. The hardware in [33] has the minimal latency, but this architecture has poor clock frequency and memory utilization. The architecture includes four memories, and their depth equals to the DHT point size. The architecture in [34] uses one true dual port RAM with depth equals to the DHT point size. One port of the RAM is used for reading and another for writing, so the latency of one time iteration of DHT equals to the DHT point size. In [35] the PE is not working in pipelined mode, so the latency is longer than our work. In our work, The latency of pipelined floating-point adder and multiplier, in our design, are $T_{add} = 7$ and $T_{mul} = 7$. The latency of the pipelined butterfly PE is $2T_{add} + T_{mul} = 21$. The address generator needs 2 clock cycles to make an address pair before a new iteration in DHT SFG. When sampling length is 1024, the proposed DHT SFG has 19 iterations after optimization. In one stage iteration, all data read to the SPFP PE requires 512 clock cycles. The last IFFT iteration has no multiplications. So the latency in our proposed DHT processor is $(21+2+512)\times 19 - T_{mul} - T_{add} = 10151$. In Table 3, our work has the minimal latency for DHT computation with point size equals to 1024.

5 Conclusion

A configurable SPFP DHT processor is proposed to accelerate the calculation of filter in Katsevich algorithm. The processor is of memory based architecture with one pipelined butterfly PE and supports variable point sizes from 8 to 1024. The DHT processor is controlled by the address

TABLE 3
PERFORMANCE COMPARISON OF DIFFERENT MULTIPLIER ARCHITECTURES

Designs	Point sizes	Clock rate (*MHz*)	FFT latency	FFT time (*us*)	DHT latency	Discrete HT time (us)	REGs	LUTs	DSPs	FPGA
[33] FFT	1024	150	5220	34.8	10752	70.2	-	-	-	XC2V1000-6
[34] Altera	1024	185	10630	57.45	21261	114.92	-	-	-	Altera
[35] Xilinx	1024	374	9427	25.20	18855	50.414	1987	2028	6	XC5VLX50T-2
Our work	1024	335	5336	15.92	10151	30.30	5368	10118	0	XC5VLX50T-2

generator. According to the point size, the address generator yields one memory address pair per clock cycle to keep the processor accessing memories successively. The DHT is calculated easily via complex multiplications in the frequency domain. Two FFTs are required in the entire process. The radix-2 FFT algorithm with DIF decomposition is utilized in the design to construct an efficiently SFG for DHT calculation. Arithmetic calculations, in the last FFT iteration, complex multiplications and the first IFFT iteration are replaced with conjugation and swapping operations, so two iterations are saved in the DHT SFG. Data are loaded and unloaded simultaneously after one frame data calculation is completed. The symmetric property of twiddle factors is utilized to decrease half size of the ROM. Truncation is used in the design to reduce data path width. The proposed DHT processor is written in Verilog HDL, so it is easy for ASIC implementation. Compared with previous works, the performance analysis shows that the proposed DHT processor has minimum clock latency.

References:

[1] A. Katsevich, Analysis of an exact inversion formula for spiral cone-beam CT, *Physics in Medicine and Biology*, Vol.47, No.15, 2002, pp. 2583-2598.

[2] A. Katsevich. Theoretically exact filtered backprojection-type inversion algorithm for spiral CT, *SIAM Journal of Applied Mathematics*, Vol.62, No.6, 2002, pp2012-2026.

[3] A. Katsevich, An improved exact filtered backprojection algorithm for spiral computed tomography, *Advances in Applied Mathematics*, Vol.32, No.4, 2004, pp.681-697.

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

[5] V. Cizek, Discrete Hilbert transform, *IEEE Transactions on Audio and electroacoustics*, Vol.18, No.4, 1970, pp.340-343.

[6] A.V. Oppenheim, R.W. Schafer, *Discrete time signal processing*, Prentice Hall, 1989, pp.775-810.

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

[8] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complexes Fourier series, *Mathematics of Computation*, Vol.19, No.90, 1965, pp.297-301.

[9] S. Winograd, On computing the DFT, *Mathematics of Computation*, Vol.32, No.141, 1986, pp.175-199.

[10] D. Kolba, T. Parks, A prime factor algorithm using high-speed convolution, *IEEE Trans. Acoustics, Speech Signal Process*, Vol.25, No.4, 1977, pp.281-294.

[11] 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.

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

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

[14] S. He, M. Torkelson, Designing pipeline FFT processor for OFDM (de)modulation, in *Proc. IEEE ISSSE*, 1998, pp.257-262.

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

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

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

[18] Y.T. Lin, P.Y. Tsai and T.D. Chiueh, Low-power variable-length fast Fourier transform processor, in *Proc. IEEE Computers and Digital Techniques*, Vol.152, No.4, 2005, pp.499-506.

[19] S. Minhyeok, L. Hanho, A high-speed Four-parallel radix-2^4 FFT/IFFT processor for UWB applications, in *Proc. IEEE ISCAS*, 2008, pp.960-963.

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

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

[22] 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.

[23] Y. Chu, L. Yiting, Y. Maohsu, H. Paoann, et al, A low-power 64-point pipeline FFT/IFFT

processor, *IEEE Trans. Consumer Electronics*, Vol.57, No.1, 2011, pp.40-45.

[24] IEEE standard for binary floating-point arithmetic, *IEEE Standard 754-1985*, 1985

[25] D. Cohen, Simplified control of FFT hardware, *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol.24, No.6, 1976, pp.577-579.

[26] M.C. Pease, Organization of large scale Fourier processors, *Journal of the ACM*, Vol.16, No.2, 1969, pp.474-482.

[27] A.M. Nielsen, D.W. Matula, C.N. Lyu. An IEEE compliant floating-point adder that conforms with the pipelined packet-forwarding paradigm, *IEEE Trans. Computers*, Vol.49, No.1, 2000, pp. 33-47.

[28] A. Malik, C. Dongdong, C. Younhee, et al, Design tradeoff analysis of floating-point adders in FPGAs, *IEEE Trans. Electrical and Computer Engineering*, Vol.33, No.3, 2008, pp.169-175.

[29] C. Tsen, Hardware design of a binary integer decimal-based floating-point adder, in *Proc. IEEE Computer Design*, 2007, pp.288-295.

[30] D. Tan C.E. Lemonds, M.J. Schulte. Low-power multiple-precision iterative floating-point multiplier with SIMD support, *IEEE Trans. Computers*, Vol.58, No.2, 2009, pp.175-187.

[31] S.V. Siddamal, R.M. Banakar, B.C. Jinaga, Design of high-speed floating point multiplier, *Electronic Design, Test and Applications*, 2008, pp.285-289.

[32] L.S.A. Hamid, K. Shehata, H. El-Ghitani, et al, Design of generic floating point multiplier and adder/subtractor units, in *Proc. IEEE UKSim*, 2010, pp.615-618.

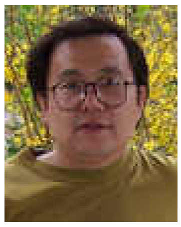
[33] M. Shengmei, Y. Xiaodong, Design of a high-speed FPGA-based 32-bit floating-point FFT processor, in *Proc. IEEE SNPD*, 2007, pp. 84-87.

[34] Altera Inc. [Online]. Available: http://www.altera.com.cn/literature/wp/wp_fft_radix2.pdf.

[35] Xilinx Inc. [Online]. Available: 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.



Wang Fei, got his Bachelor and Master degree on Automatic Control from Shandong University in 1989, and Beijing University of Aeronautics in 1994 respectively, and Ph.D. on Computer Engineering from Wright State University, OH, USA in 2006. He is currently a faculty with Harbin Institute of Technology Shenzhen Graduate School, China, and his research interest focuses on embedded system, CAD algorithms of VLSI, computer vision and so on.

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.