A Low Power ASIP for Precision Configurable FFT Processing

Yifan Bo, Jun Han†, Yao Zou and XiaoyangZeng
State-Key Lab of ASIC and System, Fudan University, Shanghai, 200433, China.
†E-mail: junhan@fudan.edu.cn

***Abstract*—Fast Fourier transformation (FFT) is a key operation in digital communication systems. Different communication standards require various FFT length and precision. In this paper, we present a low power Application-Specific Instruction-set Processor (ASIP) for variable length (16-point-4096-point) and bit precision (8-bit - 16-bit) to meet different requirements. We use scalable multipliers to construct the butterfly unit, which support both 8-bit and16-bit operation. The order of butterfly operation is adjusted to reduce twiddle-factorROM accesses, so as to reduce overall power consumption efficiently. Clock Gating is implemented to shut down processor's pipeline during the FFT processin terms of special low power demands.Special Instructions are tailored to make full use of the flexible hardware.**

I. INTRODUCTION

Recently, researches on WBAN (Wireless Body Area Network) [1] have attracted more and more attention. In a WBAN system, it will enable wireless communication between several body sensor units (BSU) and a single body central unit (BCU) worn at the human body [2]. Among WBAN technologies, the fast Fourier transform is used for bio-signal detection, which converts signal from the time domain to the frequency domain. And it is the most power consuming task. Therefore, the FFT processing is facing low power and flexibility requirements in WBAN-based systems.

The Cooley-Tukey algorithm [3] has been widely used for FFT computation. For a N-point FFT using radix-4 decimation in time algorithm, the primary computation is the butterfly calculation, as shown in Fig.1, which performs complex multiplication and addition among the input data and twiddle factors. The radix-4 implementation requires fewer complex multiplications, but it can't support 2^n-point FFT when n is an odd number. Therefore, a mixed-radix FFT algorithm is used to solve this problem [4].

In this paper, we propose a low power, precision configurable FFT ASIP which can balance the tradeoff between flexibility and performance. We mainly focus on solutions to reduce power consumption. The rest of this paper is organized as follows. Section II describes the mixed-radix algorithm and modified control flow. Section III presents the overall architecture of the proposed FFT ASIP. Section IV introduces the FFT accelerator, including configurable butterfly unit, address generation unit and customized data memory. Section V shows the experimental results. Finally, Section VI draws conclusions.

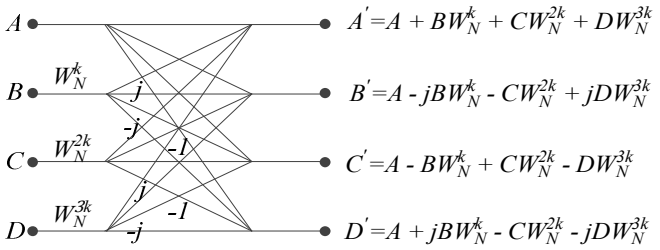

Fig. 1 A radix-4 DIT butterfly

II. MIXED-RADIX FFT ALGORITHM AND CONTROL FLOW

Radix-4 implementation can only calculate 4^n-point FFT processing. As a result, radix-2 implementation must be taken to cover all 2^n-point FFT. However, the butterfly unit of radix-4 method as seen in Fig.1 can be adjusted to provide both radix-2 and radix-4 calculation, as shown in Fig.2, which is widely used in previous work [4] [5]. With the modified radix-2/4 butterfly unit, the mixed-radix algorithm can be applied to all 2^n-point FFT just as radix-2 algorithm, while the former reducesthenumber of iterations, that is to say, fewer complex multiplications and memory accesses for memory-based architectures.

In the traditional FFT processing, the computation is divided into Stages(S), Groups(G), and Butterfly Numbers (BN). The following code shows the control flow of FFT processing.

```
FOR s = 0 to S-1
FOR g = 0 to G-1
FOR bn = 0 to BN-1
Load_Twiddle-Factors(s,bn);
Load_Data(s,g,bn);
 Butterfly;
Store_Data(s,g,bn);
      END
      END
END
```

The above computation flow requires twiddle-factor ROM access every butterfly calculation, which causes great power consumption. The function of *Load_Twiddle-Factors* only depends on the control variables *s* and *bn*. So interchanging the second loop and the third loop can reduce twiddle-factor ROM accesses dramatically. The modified control flow is as follows.

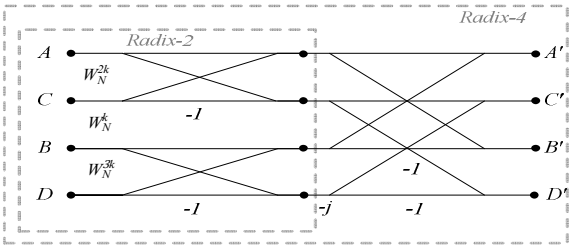

Fig. 2 A modified radix-2/4 DIT butterfly

```
FOR s = 0 to S-1
    FOR bn = 0 to BN-1
Load_Twiddle-Factors(s,bn);
            FOR g = 0 to G-1
Load_Data(s,g,bn);
                Butterfly;
Store_Data(s,g,bn);
  END
    END
 END
```

As mentioned above, the mixed-radix algorithm with a modified radix-2/4 butterfly can lower the overall power consumption due to fewer computation than the radix-2 algorithm. Moreover, the modified control flow help reduce the twiddle-factor ROM accesses, which lower the power consumption significantly.

III. ASIP Architecture and Instructions

Fig.3 shows the proposed ASIP architecture. It is a 5-stage RISC-like processor with Instruction-Fetch, Instruction-Decode, Execute/Memory access, Align and Write Back. The instruction bit-width is 16-bit in order to minimize power consumption. The register bit-width is 32-bit for loading the FFT complex products.

A FFT accelerator is employed to improve the performance of FFT processing. Meanwhile, special instructions are extended to make full use of the dedicated FFT unit. The FFT-related instructions are multi-cycle ones because they calculate a stage of FFT processing each instruction. The specific number of cycles is determined by overall points. Due to this feature, one FFT-related instruction will take a lot of cycles while other parts of the ASIP may just stall. Therefore, clock gating is implemented to cut the dynamic power of ASIP excluding FFT accelerator during the FFT processing. The "finish" signal of the FFT accelerator can enable or disable the clock gating unit.

There are four FFT-related instructions. All of them are SIMD ones, as shown in Table I. The instructions with "FFT16" prefix can compute 32-bit complex data combining 16-bit real with 16-bit imaginary data. In contrast, the "FFT8" prefixed instructions deal with 16-bit complex data, whose real and imaginary data are both 8-bit. The postfix "gated" means implementing clock gating to shut down the processor's pipeline while doing FFT processing. And the postfix "ungated" allows the processor to execute following

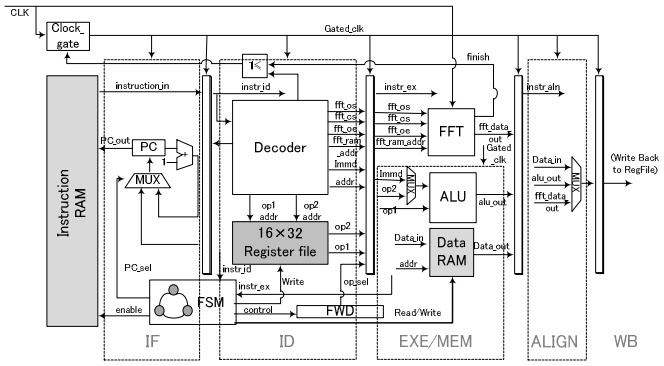

Fig. 3 ASIP architecture

programs other than FFT-related instructions, such as ALU, Load/Store instructions.

IV. FFT Accelerator

The FFT accelerator computes transforms from 16-point to 4096-point complex inputs. Bit-precision scaling is implemented: 16-bit precision mode for 16-point to 4096-point or 8-bit precision mode for 16-point to 64-point. Fig.4 illustrates the FFT accelerator's architecture including address generation unit (AGU), Twiddle ROM, DATA Memory and Butterfly Unit.

A. Configurable Butterfly Unit

The butterfly unit, as shown in Fig.5, is designed according to the following equations.

$$A' = \left(A+CW_N^{2k}\right)+\left(BW_N^{k}+DW_N^{3k}\right)$$
$$B' = \left(A-CW_N^{2k}\right)-j\left(BW_N^{k}-DW_N^{3k}\right)$$

TABLE I
FFT-Related Instructions

Instruction name	Complex data bit-width	Clock gating
FFT16_gated	32-bit	enabled
FFT16_ungated	32-bit	disabled
FFT8_gated	16-bit	enabled
FFT8_ungated	16-bit	disabled

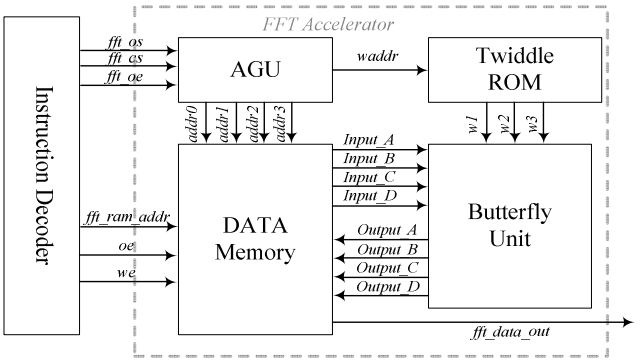

Fig. 4 FFT accelerator's architecture

$$C'=(A+CW_N^{2k})-(BW_N^{k}+DW_N^{3k})$$

$$D'=(A-CW_N^{2k})+j(BW_N^{k}-DW_N^{3k})$$

Where A,B,C and D are butterfly inputs, and W_N^k, W_N^{2k} andW_N^{3k} are twiddle factors. All the variables are complex data.

The butterfly unit consists of three complex multipliers and eight complex adders. The complex multipliers can be configured for 8-bit or 16-bit multiplication. Four multiplexers are employed to select either radix-2 calculation or radix-4 calculation. So the modified radix-2/4 butterfly unit can process either two radix-2 butterflies or one radix-4 butterfly. In order to prevent overflow errors, the butterfly unit produces 38-bit complex results for 16-bit precision mode and 22-bit complex results for 8-bit precision mode [6]. The FFT data memory stores the 38-bit or 22-bit products. Anoverflow detection mechanism is implemented to notify the memory controller how to load complex data.

B. Address Generation Unit

The address generation unit controls the data memory reading and writing operation to and from butterfly unit.It produces both data memory addresses and twiddle-factor ROM address. Firstly, it generates FFT inputs index and twiddle factors index based on the current stage counter, group counter and butterfly counter. The first stage of N-point DIT FFT has only one groups and the group has N/4 butterflies. The second stage has four groups and each group has N/16 butterflies and so forth. After the index generation, an in-place strategy, which interchanges the storage locations of the butterfly outputs, is implemented to avoid memory bank conflicts[7]. So the four inputs can be read concurrently and the four outputs can be writtensimultaneously. The memory address and bank number of 2^n-point FFT can be expressed as follows.

$$Address = b[n-1]b[n-2]b[n-3]\cdots b[4]b[3]b[2]$$

$$Bank = \{b[n-1]b[n-2]\} \oplus \{b[n-3]b[n-4]\}$$

$$\oplus \cdots \oplus \{b[3]b[2]\} \oplus \{b[1]b[0]\}$$

Where $\oplus$ means the modulo-4 addition.

The special instructions are design to calculate one stage of FFT processing. Therefore, a counter logic is employed to control the computing flow. According to the modified control flow mentioned above, it calculates the first butterfly in the first group, then the first butterfly in the second group. After finishing all the first butterflies in each group, it moves on tothe second butterfly from the first group to the last group. When all the butterflies are calculated, the stage is computed and so does the current instruction.

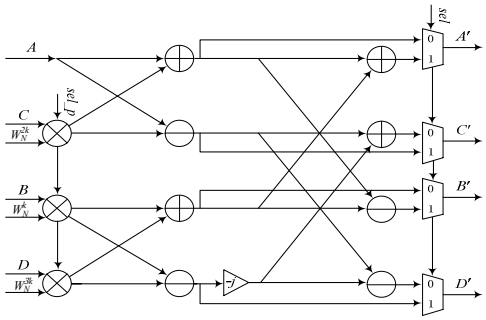

Fig. 5 Radix-2/4 butterfly unit

C. Customized FFT Data Memory

Two Ping-Pong custom memories are implemented to store the intermediate data in each stage, as shown in Fig.6. Firstly, the butterfly unit gets four complex input data from the first memory. The results are stored back to the second memory. For the next stage computation, the butterfly unit reads from the second memory and writes back to the first memory. This operation is good for increasing memory access rate.

The word length of memory is 38-bit decided by the butterfly unit, which consist 19-bit real and 19-bit imaginary data. The state of overflow flag generated in the previous stage determines which 32-bit makes up the output data [6].The complex data can be righted shifted by 1-bit, 2-bit or 3-bit toselect the [16:1], [17:2] or [18:3] bits of both the real and imaginary data. The flag is saved in order to determine the overall scaling of the results.

For the 8-bit precision mode, only 22-bit results need to be stored. So the 38-bit word length memory bank can bedivided into 22-bit and 16-bit banks. Therefore the 16-bit memory bank (shaded area on Fig.6) can be disabled for the 8-bit precision mode in order to reduce power consumption.

V. EXPERIMENTAL RESULTS

The proposed ASIP is designed in Verilog HDL and synthesized by Synopsys's Design compiler, using the standard SMIC 0.13 um technology. The ASIP achieves relatively high SQNR and low power consumption. The execution cycles and SQNR of 16-bit precision mode are presented in Table II. For the 8-bit precision mode, 16-point, 32-point and 64-point FFT are well supported. For example, the SQNR of 64-point is 27dB.

The FFT ASIP's power consumption is obtained by power simulation tool Primetime PX. The results are presented in modified mixed-radix algorithm control flow can significantly

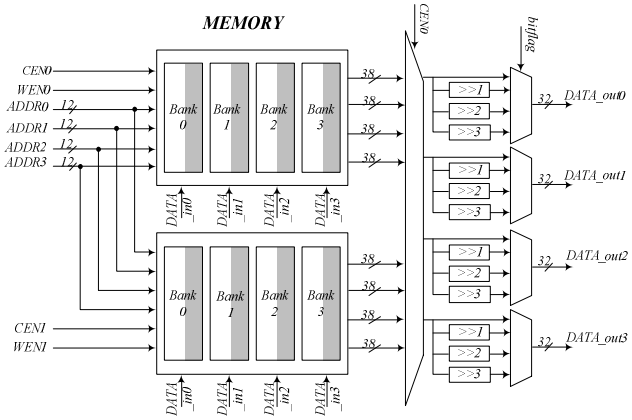

Fig. 6 Customized Ping-Pong memory

reduce twiddle-factor ROM accesses, which saves 33% power Table III. For the 16-bit precision mode, the clock gating method can cut 4% of the total power consumption. This method will be more effective if the processor concurrently execute other programs except FFT instructions. The consumption. In terms of the 8-bit precision mode, the scaling of multipliers, data memory and twiddle-factor ROM reduces power consumption 7%, 26% and 8% respectively in contrast to the 16-bit mode, and achieves 63% total reduction compared to the original FFT ASIP.

Table IV compares the proposed FFT ASIP with other previous FFT processors. Obviously, our ASIP are more flexible that can support from 16-point to 4096-point FFT computation. Moreover, the overall power consumption of our ASIP is controlled at a rather low level at 100MHz, which is better than [8] [9] [10] [11]. In order to compare with [12] [13], power consumption are normalized as follows [12]:

$$\frac{\text{FFTs}}{\text{Energy}} = \frac{\text{Technology}/0.18\ \mu\text{m}}{\text{Power} * \text{Execution Time} * 10^3}$$

Since the previous works [12] [13] are deal with 8192-point FFT, we estimate the execution cycles of 8192-point FFT for the proposed ASIP, that is 8192 / 4 * 7 = 14336 cycles. Then the normalized FFTs/Energy of our ASIP is 57.78, which is at the same level as [12]'s 55.32 and [13]'s 59.90.

VI. CONCLUSION

A low power and precision configurable FFT ASIP is proposed. It is able to calculate from 16-point to 4096-point FFT. 16-bit precision mode and 8-bit precision mode are supported. Owing to the modified control flow and clock gating method, the overall power consumption is cut to 87.2mW for 16-bit precision mode and 51.5mW for 8-bit precision mode.

TABLE II
SQNR OF FFT ASIP

FFT Size N	Cycles per FFT	SQNR (dB)
16	16	78.91
32	48	74.31
64	62	70.99
128	168	67.75
256	296	64.03
512	688	61.26
1024	1328	57.62
2048	3128	54.81
4096	6200	51.08

TABLE III
POWER CONSUMPTION OF FFT ASIP

Unit	Original ASIP (mW)	Low Power ASIP (mW)	
	16-bit	16-bit	8-bit
Processor without FFT unit	2.8	0.2	0.2
FFT RAM	18.6	15.8	9.5
Twiddle ROM	57.5	57.6	34.9
FFT unit without RAM and ROM	59.1	13.6	6.9
Total	138	87.2	51.5

TABLE IV
COMPARISON OF DIFFERENT FFT HARDWARE DESIGN

Design	Technology	FFT-length	Frequency (MHz)	Area (mm^2)	Power (mW)
[8]	130 nm	1024	100	2.96	268.0
[9]	130 nm	4096	100	1.10	102.0
[10]	130 nm	128	70	-	132.1
		256	64	-	216.9
		512	61	-	385.9
		1024	58	-	715.9
[11]	Cyclone III	64 - 2048	63	-	120.2
[12]	180 nm	8192	20	4.84	25.2
[13]	180 nm	8192	20	3.52	40.7
proposed	130 nm	16 - 4096	100	2.23	87.2[a]
					51.5[b]

[a] 16-bit precision mode

[b] 8-bit precision mode

REFERENCES

[1] IEEE 802.15. http://www.ieee802.org/15/pub/TG6.html

[2] T. O'Donovan, J. O'Donoghue, C. Sreenan, D.Sammon, P. O'Reilly andK.A. O'Connor, "A context aware wireless body area network (BAN)," *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd Int.Conf. on*, pp.1-8, April 2009

[3] R. Blahut, *Fast Algorithms for Digital Signal Processing.* Reading, MS: Addison-Wesley Publishing Company, 1985.

[4] J.H. Baek, B.S. Son, B.G. Jo, M.H. Sunwoo and S.K. Oh, "A continuous flow mixed-radix FFT architecture with an in-place algorithm," *Circuits and Syst., 2003. ISCAS '03. Proc. of the 2003 Int.Symp. on* , vol.2, no., pp. II-133- II-136 vol.2, May 2003

[5] B.G. Jo and M.H. Sunwoo, "New continuous-flow mixed-radix (CFMR) FFT Processor using novel in-place strategy," *Circuits and Syst. I: Regular Papers, IEEE Trans.on* , vol.52, no.5, pp. 911- 919, May 2005.

[6] N. Ickes. *A micropower DSP for sensor applications*, 2008 .

[7] A.T. Jacobson, D.N. Truong and B.M. Baas, "The design of a reconfigurable continuous-flow mixed-radix FFT processor," *Circuits and Syst., 2009. ISCAS 2009. IEEE Int.Symp.on*, pp.1133-1136, May 2009.

[8] Q. Zhang and N. Meng, "A low area pipelined FFT processor for OFDM-based System." in *5th Int. Conf. on (WiCom)*, 2009.

[9] A. Ferizi, B. Hoeher, M. Jung, G. Fischer and A. Koelpin, "Design and implementation of a fixed-point radix-4 FFT optimized for local positioning in wireless sensor networks," *Syst., Signals and Devices (SSD), 2012 9th Int. Multi-Conf.on* , pp.1-4, March 2012

[10] A.G. da Luz, E.A.C. da Costa andS. Ghissoni, "Reducing power consumption in FFT architectures by using heuristic-based algorithms for the ordering of the twiddle factors," *Circuits and Syst. (LASCAS), 2012 IEEE Third Latin American Symp. on* , pp.1-4, March 2012

[11] R. Netto andJ.L. Guntzel, "A high throughput configurable FFT processor for WLAN and WiMax protocols," *Programmable Logic (SPL), 2012 VIII Southern Conf. on* , pp.1-5, March 2012

[12] Yu-Wei Lin, Hsuan-Yu Liu, Chen-Yi Lee, "A dynamic scaling FFT processor for DVB-T applications," *IEEE JSSC,* vol.39, no.11, pp. 2005- 2013, Nov. 2004.

[13] Hyun-Yong Lee; In-Cheol Park; "Balanced Binary-Tree Decomposition for Area-Efficient Pipelined FFT Processing," *IEEE Trans*, *CAS,* vol.54, no.4, pp.889-900, April 2007