Low-power, high-speed FFT processor for MB-OFDM UWB application

Guixuan Liang, Danping He, Eduardo de la Torre, Teresa Riesgo
Centro de Electronica Industrial, Universidad Politecnica de Madrid, José Gutierrez Abascal, 2, 28006 Madrid, Spain

ABSTRACT

This paper presents a low-power, high-speed 4-data-path 128-point mixed-radix (radix-2 & radix-2^2) FFT processor for MB-OFDM Ultra-WideBand (UWB) systems. The processor employs the single-path delay feedback (SDF) pipelined structure for the proposed algorithm, it uses substructure-sharing multiplication units and shift-add structure other than traditional complex multipliers. Furthermore, the word lengths are properly chosen, thus the hardware costs and power consumption of the proposed FFT processor are efficiently reduced. The proposed FFT processor is verified and synthesized by using 0.13 µm CMOS technology with a supply voltage of 1.32 V. The implementation results indicate that the proposed 128-point mixed-radix FFT architecture supports a throughput rate of 1Gsample/s with lower power consumption in comparison to existing 128-point FFT architectures.

Keywords: low-power, FFT, MB-OFDM Ultra-Wideband, mixed-radix, complex multiplier

1. INTRODUCTION

UWB has recently attracted much attention as an indoor short-range high-speed wireless communication [1]. One of the most exciting characteristics of UWB is that it can support various data rates from tens of MB/s to hundreds of MB/s, thus satisfy most of the multimedia applications such as audio and video delivery. Multiband orthogonal frequency division multiplexing (MB-OFDM) is considered as the leading choice by the 802.15.3a standardization group for use in establishing a physical-layer standard for UWB communications [2]. OFDM based UWB not only has reliable high-data-rate transmission in time-dispersive or frequency-selective channels without having complex time-domain channel equalizers, but also provides high spectral efficiency [3]. Figure 1 is the UWB physical layer in MB-OFDM frame work. In the MB-OFDM systems, the FFT processor conducts total of 128 points, including 100 data tones, 12 pilot tones, 10 guard tones and 6 null tones.

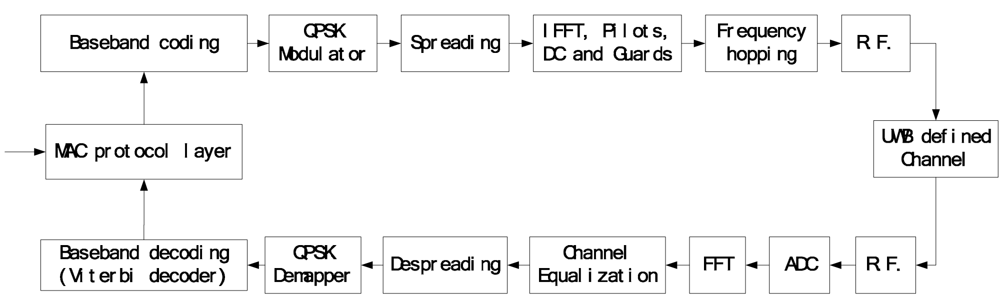

Figure 1. Multiband UWB physical layer

The FFT processor is one of the most important and complex modules in the physical layer of UWB, the execution time for 128-point should be at most 312.5 ns in order to satisfy the timing constraints. However the traditional FFT architecture cannot satisfy both the low power consumption and the high-throughput specification. Thus designers should develop high-speed power efficient FFT processor. There are many works focusing on the FFT/IFFT processor. Y. W. Lin et al. [3] proposes a mixed radix FFT algorithm by implementing radix-2 and radix-2^6 FFT. They divide the radix-2^6 FFT into two radix-2^3 FFT stages, and then realize the radix-2^3 FFT by 3 radix-2 stages. The architecture is designed as four-parallel-paths SDF pipelines, the max work frequency of FFT chip reaches 250MHz under 180nm CMOS technology library, the throughput rate is 1G sample/s with power dissipation of 175mW. Sang-In Cho et al. [4]

VLSI Circuits and Systems V, edited by Teresa Riesgo, Eduardo de la Torre-Arnanz,
Proc. of SPIE Vol. 8067, 80670E · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.890303

employs a modified radix-2^4 algorithm and a radix-2^3 algorithm to significantly reduce the number of complex constant multipliers. It also employs four-parallel-path pipelined architecture. The simulation result indicates the FFT processor can support a throughput of up to 1G sample/s with a power dissipation of 112mW under 180nm CMOS technology library. Z.Wang et al. [5] proposed a two-stage radix-2^2 and radix-32 FFT algorithm, it uses radix-2^5 algorithm instead of radix-32, four parallel pipelines are employed at the second stage. S.Qiao et al. [6] Proposes radix-2 and radix-64 FFT algorithm as [3], however, it develops a non-Cooley-Tukey radix-8 unit in order to save hardware cost. The measurement result shows the throughput rate is 409.6M sample/s with area saved by 20% to 63% compared to that of radix-2 SDF architecture. Table 1 show out the features of each algorithm.

Table 1. Feature of other proposed Algorithms

Algorithm	Y. W. Lin et al. [3]	Sang-In Cho et al. [4]	Z.Wang et al. [5]	S.Qiao et al. [6]
Architecture	Radix-2, 2 × radix-2^3	Modified radix-2^4, raidx-2^3	Two-stage radix-2^2, radix-2^5	radix-2 and radix-64
CMOS Technology Library	180nm	180nm	0.13µm & 90nm	180nm
Throughput rate (Sample/s)	1G	1G	528M	409.6M

A novel mixed radix 128-point FFT algorithm is presented in this paper and multipath pipelined 128-point FFT architecture is designed. As most of the power consumption and hardware complexity in FFT processor come from the complex multipliers, carefully design will not only lower the power, higher the speed, but also guarantee a good level of signal-to-quantization-noise ratio (SQNR).

The paper is organized as follows. Section 2 describes the novel proposed 128-point mixed-radix FFT algorithm. Section 3 introduces the hardware architecture designed for the proposed algorithm. Section 4 demonstrates experiment results and compares results with existing FFT architectures. At the end, the conclusions are drawn in Section 5.

2. 128-POINT MIXED RADIX ALGORITHM

A mixed radix-2 and radix-2^2 128-point FFT algorithm is proposed and introduced in this section. An N-point discrete Fourier transform (DFT) is defined as

$$X(k)=\sum_{n=0}^{N}x(n)W_N^{kn} \qquad k=0...N-1 \tag{1}$$

Where x(n) and X(k) are complex values. The twiddle factor is expressed in (2)

$$W_N^{nk}=e^{-j(2\pi nk/N)} \qquad W_N^{nk}=e^{-j(2\pi nk/N)} \tag{2}$$

To drive the proposed algorithm, n and k are determined by a four-dimensional linear index map:

$$n=64n_1+16n_2+4n_3+n_4 \qquad \begin{cases} n_1=0,1 \\ n_2,n_3,n_4=0,1,2,3 \end{cases} \tag{3}$$

$$k=32k_4+8k_3+2k_2+k_1 \qquad \begin{cases} k_1=0,1 \\ k_2,k_3,k_4=0,1,2,3 \end{cases} \tag{4}$$

Substitute (3) and (4) into (1), we obtain (5) as follow

$$X(k)=\sum_{n=0}^{N}x(64n_1+16n_2+4n_3+n_4)W_2^{n_1k_1}W_{128}^{k_1(16n_2+4n_3+n_4)}W_4^{n_2k_2}W_{64}^{k_2(4n_3+n_4)}W_{16}^{n_4k_3}W_4^{n_4k_4} \tag{5}$$

As we can see in (5), the 128-point FFT is changed into one radix-2 and three radix-4 stages. Therefore, we have stage1 expressed by $S_1(n_2, n_3, n_4, k_1)$, which contains a radix-2 algorithm and a multiplication of twiddle factor $W_{128}^{k_1(16n_2+4n_3+n_4)}$, as shown in (6).

$$S_1(n_2, n_3, n_4, k_1) = \sum_{n_1=0}^{1} x(64n_1 + 16n_2 + 4n_3 + n_4) \cdot W_2^{n_1 k_1} W_{128}^{k_1(16n_2+4n_3+n_4)} \tag{6}$$

Further decompose n_2 and k_2 in (7) and (8), stage2 can be represented by two radix-2 butterflies multiplied by twiddle factor $W_{64}^{k_2(4n_3+n_4)}$ as shown in (9).

$$n_2 = 2\alpha_1 + \alpha_2 \qquad \alpha_1, \alpha_2 = 0,1 \tag{7}$$

$$k_2 = 2\beta_2 + \beta_1 \qquad \beta_1, \beta_2 = 0,1 \tag{8}$$

$$S_2(n_3, n_4, k_1, k_2) = \sum_{\alpha_2=0}^{1} \sum_{\alpha_1=0}^{1} S_1(2\alpha_1 + \alpha_2, n_3, n_4, k_1) \cdot W_2^{\alpha_1 \beta_1} W_4^{\alpha_2 \beta_1} W_2^{\alpha_2 \beta_2} W_{64}^{k_2(4n_3+n_4)} \tag{9}$$

By using similar method, stage three and four can be represented as (10) and (11) respectively. Figure 2 is the signal flow of the proposed FFT algorithm.

$$S_3(n_4, k_1, k_2, k_3) = \sum_{\alpha_4=0}^{1} \sum_{\alpha_3=0}^{1} S_2(2\alpha_3 + \alpha_4, n_4, k_1, k_2) \cdot W_2^{\alpha_3 \beta_3} W_4^{\alpha_4 \beta_3} W_2^{\alpha_4 \beta_4} W_{16}^{k_3 n_4} \tag{10}$$

$$X(k) = \sum_{\alpha_6=0}^{1} \sum_{\alpha_5=0}^{1} S_3(2\alpha_5 + \alpha_6, k_1, k_2, k_3) \cdot W_2^{\alpha_5 \beta_5} W_4^{\alpha_6 \beta_5} W_2^{\alpha_6 n_6} \tag{11}$$

With

$$\begin{cases} \alpha_3, \alpha_4, \alpha_5, \alpha_6 = 0,1 \\ \beta_3, \beta_4, \beta_5, \beta_6 = 0,1 \end{cases} \tag{12}$$

Note that the inverse FFT (IFFT) of a length-N complex sequence can be obtained by (13)

$$x(n) = \frac{1}{N} \left(\sum_{k=0}^{N-1} X^*(k) W^{kn} \right)^* \tag{13}$$

The IFFT can be realized by making the complex conjugate at the input and without changing coefficients, then take the conjugate at the output and divide the result by N.

3. ARCHITECTURE OF THE PROPOSED FFT ALGORITHM

According to the proposed algorithm, a 4-data-path SDF pipeline processor was proposed. The block diagram of the proposed architecture is given in Figure 3. Module one to four represents to stage one to four in signal flow respectively. The function of Module 1 is to realize the radix-2 FFT algorithm, while Modules 2 to 4 realize the radix-2^2 FFT algorithm. We defined 3 types of word lengths for our system: input/output word-length (IOWL), system word-length (SWL) and twiddle word-length (TWL). In this paper, we choose IOWL =SWL = 10 bits, TWL = 8 bits, the details will be discussed in Section 4.

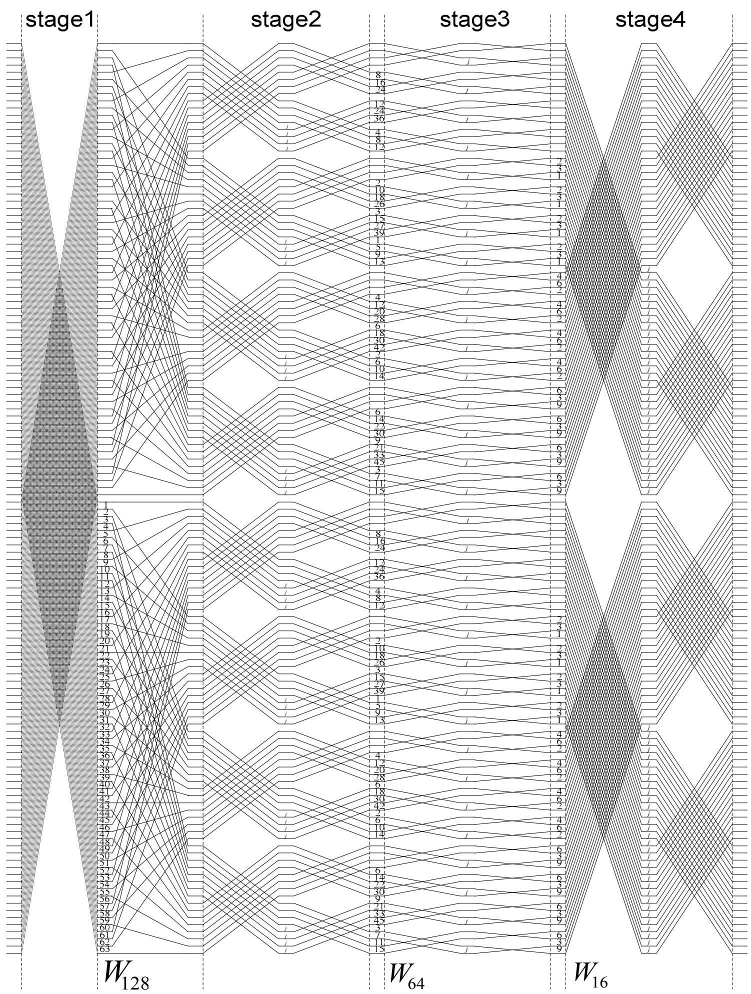

Figure 2 . Signal Flow of proposed FFT algorithm

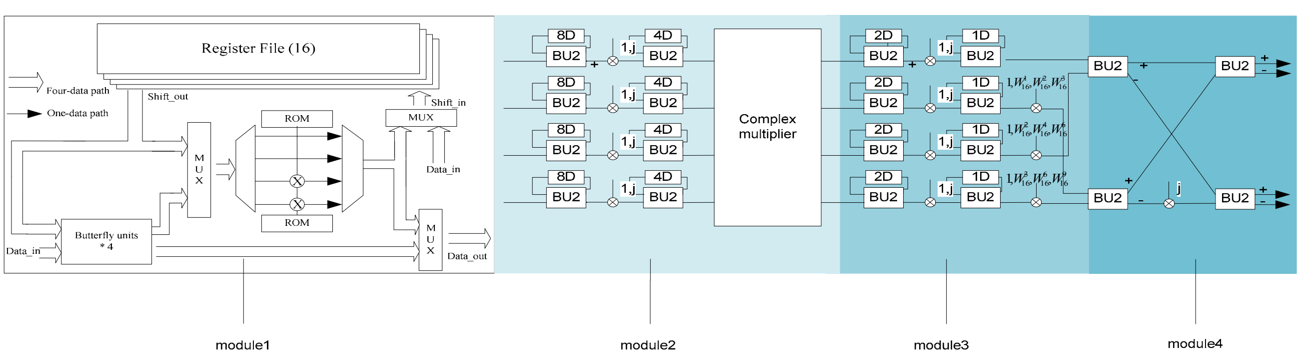

Figure 3. Block diagram of proposed four-parallel data-path 128-point mix-radix FFT processor

Module 1 consists of four register files (each can store 16 pieces of complex data), two complex Booth's multipliers, four radix-2 butterfly units (BU_2), two ROMs and some multiplexers. The function of ROM is used to store TWs for $W_{128}^{k_1(16n_2+4n_3+n_4)}$. By using the periodical property of twiddle factors (TW), only 1/8 periods of the cosine and sine waveforms are stored in ROM. At the first 16 cycles, the first 64 input data are stored in the register file. Then from the 17th cycle, the BUs begin to execute the complex addition and subtraction between x(n) and x(n+64) during the next 16 clock cycles. The results of addition (x(i)) will be fed to Module 2 directly as no multiplications are needed, while the results of subtraction (y(i)) are multiplied by TW and then stored into the register file before they are sent to Module 2. Generally speaking, four complex multipliers are needed in the four-parallel approach to implement the radix-2 FFT algorithm, thus the utilization rate of the complex multiplier is only 50%. While in our proposed architecture, as Module 2 needs 32 clock cycles to process x(i), we can share the complex multiplier for four paths separately during these 32 cycles. The detailed operation is described below. When y(i) are generated from BUs, two of the y(i), y(0) and y(1), are multiplied by the appropriated twiddle factors first while y(2) and y(3) are going to the register files. 16 clock cycles later, other two, y(2) and y(3), are multiplied then be fed to Module 2 at the same time with the results of y(0) and y(1). By using this rescheduling architecture, only two complex multipliers are needed, thus 50% of multipliers are saved and a 100% utilization of the multipliers is achieved.

Module 2 consists of four-parallel radix-2^2 SDF architectures and a complex multiplier module for W_{64}^{nk} as shown in Figure 4. As can be seen in Figure 3, the output data generated by the BU2 between the first step and second step should be multiplied by j, which can be implemented efficiently by just exchanging the real part and imaginary part with each other. In order to simplify the complexity of the complex multipliers, we do a further modification for the approach proposed in [3]. The twiddle factors of the modified complex multiplier are $W_{64}^{p} = \exp(-\frac{j2\pi p}{64}) = X_p - jY_p$, where $X_p = \cos(2\pi p/64)$, $Y_p = \sin(2\pi p/64)$, p is from 0 to 45. The twiddle factor of 64 can be divided into eight regions as shown in Figure 5. Region A consists of values with p from 0~8, the values in other seven regions can be represented by transforming the data of region A according to Table 2. Therefore, through the mapping method, only nine sets of constant values are needed. In practice, we only need to implement eight sets of constant values in region A, since the first pair of constant values (1, 0) is trivial. In addition, these constant values can be realized more efficiently by using 8-bit shift-add multiplier [7].

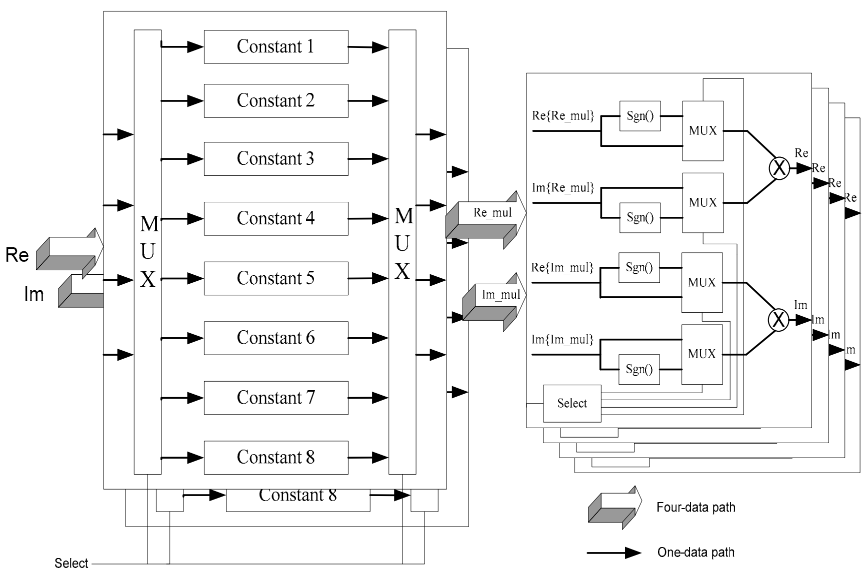

Figure 4. Modified Complex Multiplier

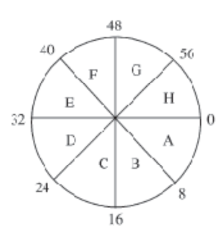

Figure 5. Twiddle factor of 64 can be divided into eight regions

Table 2. Mapping table for twiddle factor in different region

Region	Real	Imaginary
A (p=x*)	X_p	$-Y_p$
B (p=16-x)	Y_p	$-X_p$
C (p=x-16)	$-Y_p$	$-X_p$
D (p=32-x)	$-X_p$	$-Y_p$
E (p=x-32)	$-X_p$	Y_p
F (p=48-x)	$-Y_p$	X_p

Table 3 shows the schedule of the twiddles for the four paths. The table only shows 16 clock cycles because the values of twiddle factor are repeated every 16 cycles. After mapping according to Table 2, the results of the coefficient value and corresponding regions are shown in Table 4 and Table 5. It can be clearly seen from Table 4 that the twiddle factor of four paths in each time slot has different values, except for the first 4 cycles which no multiplications are needed. As can be seen from the block diagram of the complex multiplier in Figure 4, when the inputs come in the first module, the four path data are mapped into different complex constant multipliers according to the schedule in Table 4. After the multiplications, the regions are selected at second module according to Table 5 and Table 2. By using the modified complex multiplier, the system is much simpler, efficient in time and energy as only few registers and adders are needed.

Table 3. Scheduling of the twiddle factor, W_{64}^{x}, where x is the value shown in the table

L1	0	0	0	0	0	8	16	24	0	12	24	36	0	4	8	12
L2	0	0	0	0	2	10	18	26	3	15	27	39	1	5	9	13
L3	0	0	0	0	4	12	20	28	6	18	30	42	2	6	10	14
L4	0	0	0	0	6	14	22	30	9	21	33	45	3	7	11	15

Table 4. Scheduling of the twiddle factor after mapping

L1	0	0	0	0	0	8	0	8	0	4	8	4	0	4	8	4
L2	0	0	0	0	2	6	2	6	3	1	5	7	1	5	7	3
L3	0	0	0	0	4	4	4	4	6	2	2	6	2	6	6	2
L4	0	0	0	0	6	2	6	2	7	5	1	3	3	7	5	1

Table 5. Scheduling of the region after mapping

L1	A	A	A	A	A	A	B	C	A	B	C	E	A	A	A	B
L2	A	A	A	A	A	B	C	D	A	B	D	E	A	A	B	B
L3	A	A	A	A	A	B	C	D	A	C	D	F	A	A	B	B
L4	A	A	A	A	A	B	C	D	B	C	E	F	A	A	B	B

Module 3 consists of four-parallel BU_2s and only three substructure-sharing multiplication units for TW W_{16}^{nk} according to the proposed algorithm, which is one less than that in [4]. Furthermore, as the TWL of the proposed FFT is 2 bits less than that of [3] and [4], fewer additions are needed, the hardware cost is reduced, and the speed is improved.

We name the three complex multipliers in module 3 as TCM_1, TCM_2, and TCM_3. Table 6 is the time schedule of TCM_1, TCM_2 and TCM_3, the value in the table represents to p in W_{16}^{p}, the table shows that the values of the multipliers repeat every 4 cycles. Besides, $W_{16}^{0}=1$, which is trivial, therefore only 3 twiddle factors should be designed in each multipliers. According to Table 6, we can calculate the coefficients that are needed to represent all the twiddle factors in each multiplier. Table 7 is the coefficient table which shows that TCM_1 and TCM_3 require three coefficients while TCM_2 needs only one coefficient.

Table 6. Time Schedule for TCM_1, TCM_2 and TCM_3

TCM_1	0	2	3	1
TCM_2	0	4	6	2
TCM_3	0	6	9	3

Table 7. Coefficient Table

TCM_1	$\cos\frac{\pi}{4}, \cos\frac{\pi}{8}, \sin\frac{\pi}{8}$
TCM_2	$\cos\frac{\pi}{4}$
TCM_3	$\cos\frac{\pi}{4}, \cos\frac{\pi}{8}, \sin\frac{\pi}{8}$

The architecture of TCM1 is shown in Figure 6. TCM1_S1 controls whether to exchange the real part with the imaginary part at the input, TWD1 in Figure 6 is the coefficient multiplier which multiplies the input by $\cos\frac{\pi}{4}, \cos\frac{\pi}{8}, \sin\frac{\pi}{8}$ at the same time, the three outputs are selected by control signal TCM1_S2. Finally, the computed results are selected by TCM1_S3. TCM2 (Figure 7) and TCM3 (Figure 8) have similar structure with TCM1, only with different time schedule.

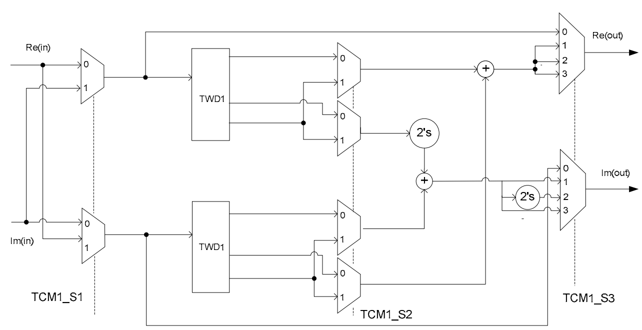

Figure 6. Architecture of TCM_1

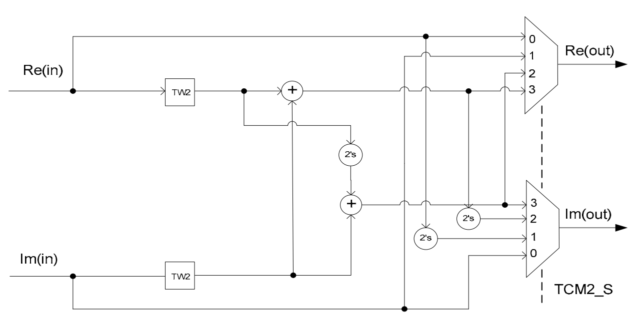

Figure 7. Architecture of TCM_2

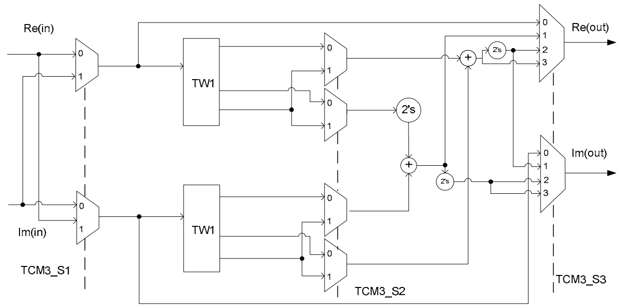

Figure 8. Architecture of TCM_3

Table 8 shows the 2's complement values of the coefficients for TWD1 in TCM_1 and TCM_3. We notice that the binary values of a, b and c have some parts in common, therefore to reduce the hardware cost, we decompose the binary values to find the sharing parts. The proposed architecture of TWD1 and TWD2 are shown in Figure 9 and Figure 10. By using the proposed shift-add method, the complex multipliers in Module 3 are even faster and less power consume compared with that proposed in Module 2. Thus, a very high speed, low-power system is realized during this stage.

Table 8. 8 bits Binary of the coefficients and decomposition (for TCM_1 and 3)

Coefficients	2's complement	2's complement decomposition
$a = \cos\frac{\pi}{8}$	01110110	00110110
		01000000
$b = \sin\frac{\pi}{8}$	00110001	00110000
		00000001
$c = \cos\frac{\pi}{4}$	01011011	00011011
		01000000

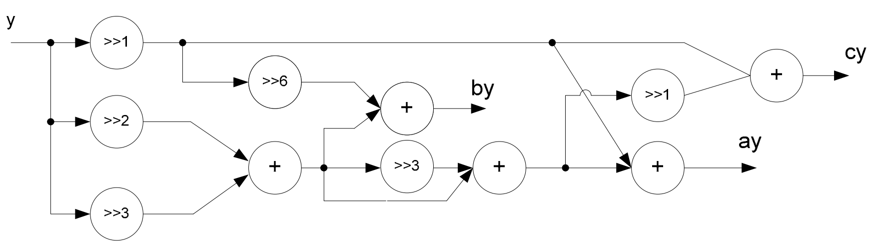

Figure 9. Architecture of TWD1

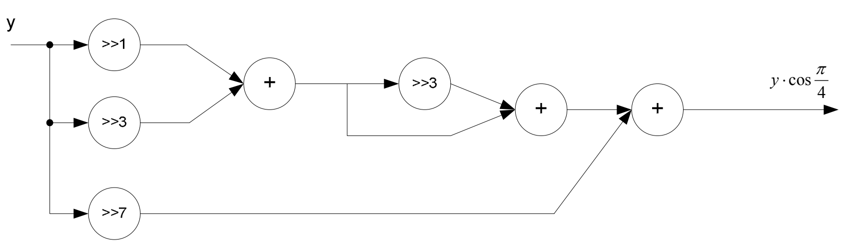

Figure 10. Architecture of TWD2

Module 4 has two stages of BU_2s, only one twiddle factor of 'j' is needed which can be realized by exchanging the real part and imaginary part of the complex data.

4. EXPERIMENTAL RESULTS

Before implementing the hardware module, the proposed architecture is verified in Simulink. In order to choose properly the bit length for the system, we employ fix-point tool box to estimate the relationship between the Signal to Quantization Noise Ratio (SQNR) of the outputs and different IOWLs, SWLs and TWLs. As the word lengths increase, the average output SQNR will increase, while the hardware cost will increase as well. Thus choose the word length is a trade-off problem. We employ fix point tool box in MATLAB to simulate the affect of word length on GR. Three types of word lengths are defined: input/output word-length (IOWL), system word-length (SWL) and twiddle word-length (TWL). [8] proves that SQNR of FFT module in UWB should be enough when IOWL=6bits, SWL=TWL= 11bits. We also notice that in [4] the SQNR reaches 35 dB with SWL=TWL=IOWL=10 bit.

We simulate 18 groups of test data by implement different word lengths on GR, Table 9 shows the results. As the SQNR reaches 27.4dB when the word lengths are the same as [8], similarly as [4], the average SQNR is 36.7dB when bit length equals to 10. As aforementioned, besides the design of HW architecture, the word-length decision helps to reduce the power consumption. Thus to be reasonable, for instance we keep the IOWL and SWL as 10 bits, while reduce the TWL to 8 bits to slightly lower the SQNR, as see in Figure 11, the average SQNR is 34.1 dB which still satisfy the requirement of MB-OFDM UWB protocol.

Table 9. Output SQNR summary table

IOWL,SWL,TWL(bit)	10,10,10	6,11,11	10,10,8
Worst output SQNR	35.6	26.6	33.3
Best output SQNR	38.4	27.9	34.7
Average SQNR	36.7	27.4	34.1

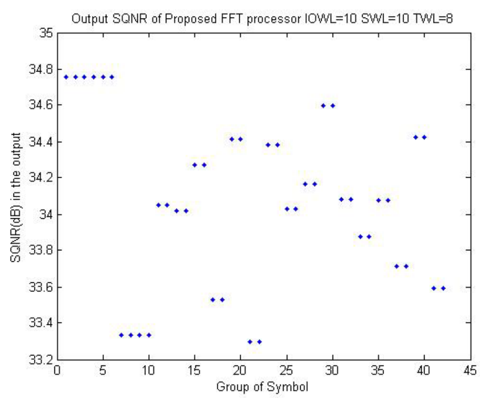

Figure 11. SQNR of proposed FFT processor (IOWL=SWL=10bits, TWL=8bits)

After that, a manually optimized implementation in Verilog HDL of the proposed FFT processor has been obtained. The hardware module is verified and the outputs SQNRs of different input data are calculated, the average SQNR is 34dB.

At the end, the proposed FFT architecture is synthesized by using the UMCL130E 0.13 μm CMOS technology with a supply voltage of 1.32V. Table 10 compares the implementation results of the proposed FFT processor with the existing works. It indicates that the proposed FFT processor can support a data processing rate of 1 G sample/s with power dissipation of 43.79 mW at 250 MHz. In order to compare the power consumption with [3] and [4], which use 0.18μm technology, the power consumption might be multiplied by a factor of around 1.4, thus is 61 mW, which is around 50% of that of [4].

Table 11 compares the hardware cost, FFT algorithm and throughput rate with two existing 128-point four-parallel data-path FFT architectures. During the test of circuit, we find that the critical path of the FFT processor lies in the two multipliers at stage 1, we employ Booth's multiplier, thus, the critical path is shortened and hardware cost of these multipliers is 75% of [3]. In the proposed architecture, only 3 trivial multipliers are needed, and their complexities are reduced because of the properties of the data stream and the sharing structure. As a result the hardware cost of the three trivial multipliers is 40% of [4] and 63.7% of [3]. The complex multiplier in stage 2 not only uses less constant complex multipliers than that of [3], but the bit-length of the TW is less as well.

Table 10. Comparison synthesis results of the FFT processor

Algorithm	Proposed algorithm	Y. W. Lin et al. [3]	Sang-In Cho et al. [4]
CMOS Technology Library	130nm	180nm	180nm
Working Frequency(MHz)	250	250	250
Equivalent 2*1Nand Gate Count	80,424	N/A	80,100
Dynamic Power /mW	43.79	175	112

Table 11. Comparison synthesis results of the FFT processor

Algorithm	Proposed algorithm	Y. W. Lin et al. [3]	Sang-In Cho et al. [4]
Architecture	Radix-2, 2 × radix-2^2	Radix-2, 2 × radix-2^3	Modified radix-2^4, raidx-2^3
No. of complex registers	124	124	124
No. of nontrivial multipliers	$4\times0.55+2\times0.58$	$2+4\times0.62$	4×0.6
No. of trivial multipliers	$2\times1.5+1\times0.82$	6	$4\times1.97+2\times0.82$
No. of complex adders	48	48	48
Throughput rate (R: clock rate)	4R	4R	4R

5. CONCLUSIONS

In this paper, we propose a novel mixed radix 128-point FFT algorithm by combining modified radix-2 and radix-2^2 algorithms together. A low-power, high throughput rate FFT processor is built. Thanks to the algorithm, we are able to significantly reduce the number and complexity of the complex multipliers. The power consumption of the whole system has been reduced by around 50% compared with that of existing work. The implementation results indicate that the throughput rate of the proposed FFT processor with 10-bit IOWL, SWL and 8-bit twiddle factor word-length can support 1 Gsample/s with a power consumption of 43.79 mW at 250 MHz by using 0.13 μm CMOS technology.

We are currently finishing the design of the other modules of UWB system and, at the same time, refining the design method to assure first-silicon access.

REFERENCES

[1] M. B. Blanton, “An FPGA Software-Defined Ultra Wideband Transceiver,” M.S. Thesis, Virginia Polytechnic Institute and State University, (2006).

[2] A. Batra et al., “Multi-Band OFDM Physical Layer Proposal for IEEE 802.15 Task Group 3a,” IEEE P802.15-03/268r3, (2004).

[3] Y. W. Lin, H. Liu and C. Lee, “A 1-GS/s FFT/IFFT Processor for UWB Applications,” IEEE Journal of solid-state circuits, (2005).

[4] S, Cho and K. Kang, “A Low-Complexity 128-point Mixed-Radix FFT Processor for MB-OFDM UWB Systems,” ETRI Journal, (2010).

[5] Z.Wang et al., “A Novel FFT processor for OFDM UWB systems”, Proc. IEEE APCCAS, Dec.2006, pp. 374-377.

[6] S.Qiao, Y.Hei, B. Wu, Y. Zhou “An Area and Power Efficient FFT Processor for UWB Systems,” Proc. IEEE WICOM, 582-585 (2007).

[7] Maharatna, K., Grass, E. and Jagdhold, U., “A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM,” IEEE Journal of Solid-State Circuit, 39 (3). 484-493 (2004).

[8] R S Sherratt, S Makino, “Numerical Precision Requirements on the Multiband Ultra-Wideband System for Practical Consumer Electronic Devices,” IEEE Transactions on Consumer Electronics, Volume 51(2), 386-392 (2005).