

Short Paper

A Register Array Based Low Power FFT Processor for Speech Recognition*

GIN-DER WU AND YING LEI
Department of Electrical Engineering
National Chi Nan University
Puli, 545 Taiwan

Fast Fourier Transform (FFT) plays an important role in the field of digital signal processing. High performance FFT processors are widely used in different application, such as speech processing, image processing, and communication system. In this paper, we proposed a novel register array based low power FFT processor for Mel Frequency Cepstral Coefficient (MFCC). Compared with [9-12], this novel architecture can reduce more power consumption. This approach is very attractive for the speech feature extraction of MFCC.

Keywords: digital signal processing, FFT processor, register array, low power, Mel Frequency Cepstral Coefficient (MFCC)

1. INTRODUCTION

The feature extraction is an important issue in the field of speech recognition. There are two common speech features for speech recognition. One is Linear Predictive Coefficient Cepstrum (LPCC). The other is Mel Frequency Cepstral Coefficient (MFCC). Since MFCC can express human acoustic character more efficiently than LPCC [1], this paper will focus on MFCC. FFT is a very important component in calculating MFCC. As far as FFT is concerned, there are two popular hardware architectures to implement it. One is pipeline-based design [2], and the other is memory-based design [3]. For pipeline architecture, there are two basic hardware architectures. One is multiple-path delay commutator pipeline architecture (MDC), and the other is single-path delay feedback pipeline architecture (SDF). Radix-2 MDC (R2MDC) [4] was the most classic architecture for pipeline-based FFT design. It can provide high performance. However, the hardware architecture is quite complex, and the memory size is very large. Therefore, Radix-2 SDF (R2SDF) [5] is proposed to overcome those disadvantages. Pipeline architecture provides high performance and simple control unit. However, the bit-reversed output order needs extra buffer to reorder the output data. This architecture still needs too much hardware resource. For memory-based architecture, this architecture usually uses one butterfly unit (BF) to compute FFT. Unlike SDF and MDC architecture, it usually has one or two

Received January 5, 2007; revised April 4 & May 24, 2007; accepted June 22, 2007.
Communicated by Liang-Gee Chen.

* This work was supported by the National Science Council of Taiwan, R.O.C. under grant NSC 94-2213-E-260-010. The preliminary version of this paper has been presented in 2006 IEEE Conference on Systems of Systems Engineering, Los Angeles, CA, U.S.A.

memories to store the results computed by butterfly unit. In addition, it just needs one ROM to store the coefficients. Therefore, it needs lower cost than SDF and MDC. However, its fault is loss in throughput rate.

Some researchers investigated several issues of low power FFT processor. In Baas *et al.* [6], the processor operated at low supply voltage V_{dd} , which approaches the value of the transistor thresholds V_t , to dramatically increase the overall energy efficiency. It adopted cached-memory architecture in order to offer faster speed and lower power consumption. In [7], some researchers reduced power consumption by minimizing the number of complex multiplications. In [8], the authors designed a low power processor by using asynchronous elements. In [9], power saving was achieved by using the most appropriate FFT size instead of a fixed large size FFT for worst case channel conditions. In [10], clock gating technique was implemented to disable the memory modules which are not in use for the current FFT size in order to save power. [11] proposed a 64-point delay-balanced SRSDF FFT pipeline architecture. The SRSDF FFT pipeline can achieve a higher clock rate because of the well balance of the multiplication and the butterfly operation. [12] proposed a dynamic scaling FFT processor which supports 8k mode DVB-T system. With data scheduling and pre-fetched buffering, single-port memory can be adopted without degrading throughput rate. [13] proposed a 16-bit low energy asynchronous 128-point FFT/IFFT processor which is fabricated by $0.35\mu\text{m}$ CMOS design rule. It operates energy-critical medium-to-low speed applications at low supply voltages (1.1V~1.4V).

In this paper, we use a butterfly unit to design our FFT processor architecture. Unlike memory-based architecture, we adopt pipeline architecture to improve our hardware performance. Furthermore, we propose a novel asynchronous register array instead of memory to reduce power consumption. The advantage of the asynchronous circuit is the smaller area. For example, the area of the Flip-Flop is three times large than Latch. Furthermore, asynchronous circuit does not need clock signal, so it can reduce the power consumption of clock tree. The simulation result shows that we can save power consumption up to 84%. The proposed hardware architecture suits for low radix FFT processor.

2. FFT/IFFT HARDWARE ARCHITECTURE

The flowchart of calculating MFCC is shown in Fig. 1. FFT is a very important component to calculate MFCC. The sampling rate is 8 KHz, and the frame size is 30ms. In this paper, we implement a 256-point FFT processor which could finish executing in 30ms and has low power consumption. The FFT/IFFT hardware architecture which we proposed is radix-2 Decimation-In-Frequency (DIF) algorithm.

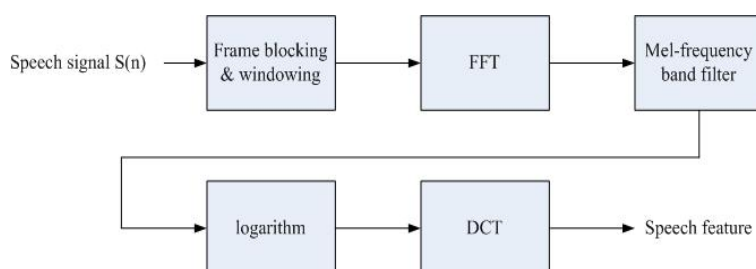


Fig. 1. The flowchart of calculating MFCC.

We depict even-number frequency samples as the following equations.

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \text{ where } W_N = e^{-j(2\pi/N)}, k = 0, 1, 2, \dots, N-1 \quad (1)$$

$$X(2r) = \sum_{n=0}^{N-1} x(n)W_N^{2rn}, \text{ where } r = 0, 1, 2, \dots, (N/2) - 1 \quad (2)$$

$$X(2r) = \sum_{n=0}^{N/2-1} x(n)W_N^{2rn} + \sum_{n=N/2}^N x(n)W_N^{2rn} \quad (3)$$

$$X(2r) = \sum_{n=0}^{N/2-1} x(n)W_N^{2rn} + \sum_{n=0}^{N/2-1} x(n + N/2)W_N^{2r(n+N/2)} \quad (4)$$

$$X(2r) = \sum_{n=0}^{N/2-1} [x(n) + x(n + N/2)]W_{N/2}^{rn} \quad (5)$$

According to the above equations, we use the same manner to obtain the odd-number samples as the following equations.

$$X(2r + 1) = \sum_{n=0}^{N-1} x(n)W_N^{n(2r+1)}, \text{ where } r = 0, 1, 2, \dots, (N/2) - 1 \quad (6)$$

$$X(2r + 1) = \sum_{n=0}^{N/2-1} [x(n) - x(n + N/2)]W_N^n W_N^{rn}, \text{ where } r = 0, 1, 2, \dots, (N/2) - 1 \quad (7)$$

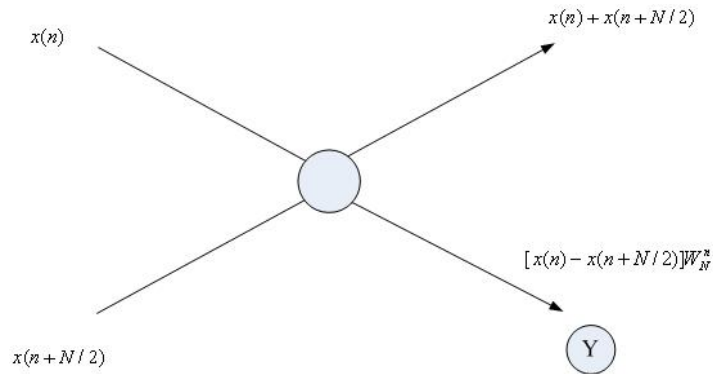


Fig. 2. The butterfly graph of the radix-2 DIF FFT.

The decomposition can be mapped to a butterfly (BF) graph which is shown in Fig. 2. In this figure, we describe the $x(n)$ and $x(n + N/2)$ as the following.

$$x(n) = ar + j \times ai, \quad (8)$$

$$x(n + N/2) = br + j \times bi, \quad (9)$$

$$x(n) - x(n + N/2) = in1 + j \times in2, \quad (10)$$

We design the BF hardware architecture in node Y by deriving the following equation.

$$(in1 + j \times in2) \times (\cos\theta + j \times \sin\theta) \quad (11)$$

$$= in1 \times \cos\theta + j(in1 \times \sin\theta) + j(in2 \times \cos\theta) - (in2 \times \sin\theta) \quad (12)$$

$$= (in1 \times \cos\theta - in2 \times \sin\theta) + j(in1 \times \sin\theta + in2 \times \cos\theta) \quad (13)$$

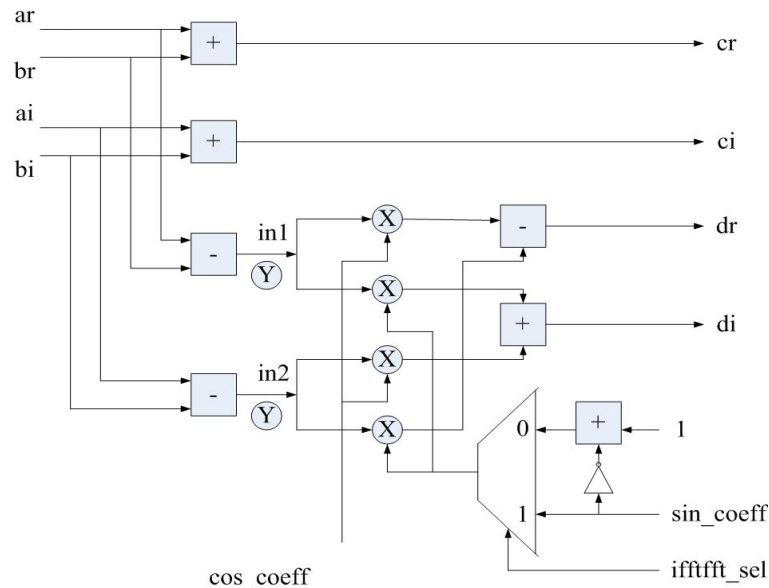


Fig. 3. The radix-2 butterfly unit (BF) architecture.

According to the above equations, the BF hardware architecture is shown in Fig. 3. It uses four multipliers, four adders, and three subtractors. Based on this BF, we use multiple-path architecture to implement the FFT processor in Fig. 4. This structure has three pipelines. The first pipeline is the select module (SE). This module can select input signal and determine what mode to execute (ifft or fft). The second pipeline is butterfly unit (BF) which bases on radix-2 algorithm. The third pipeline is register array module (RA). This module is to store the result and read the input data. In this paper, we adopt register array to store the data instead of on-chip SRAM. In this module, we adopt asynchronous design methodology to reduce power consumption.

For an example of 32-point FFT, there are 5 stages. In each stage, the butterfly unit (BF) is executed 16 times. After execution, each BF produces four outputs. Hence, we need four register banks to save the outputs. Since each stage executes BF 16 times, each bank has 16 register. The structure of the register array is shown in Fig. 5. There are two levels in this register array. The first level (Bank 0 ~ Bank 3) needs 64 registers. In addition, the second level (Bank 4 ~ Bank 7) would be saved when the front stage finished all execution. When the address generator in Fig. 4 sends the address to the register array, the second level of the register array in Fig. 5 would transmit correct data back to the BF.

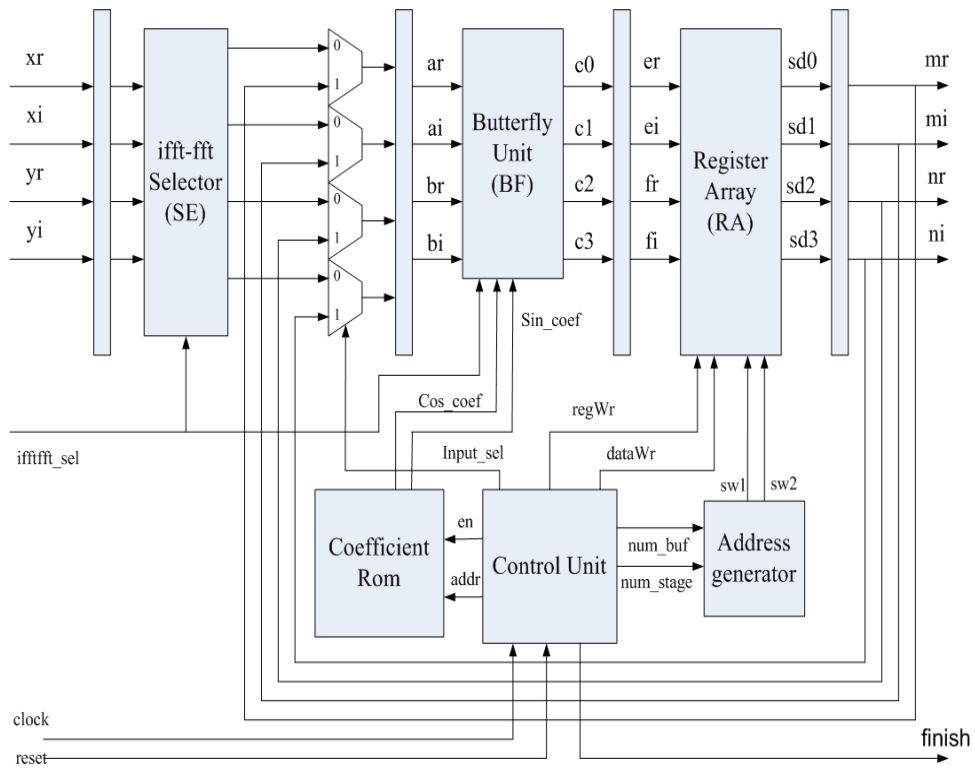


Fig. 4. The radix-2 pipeline FFT processor architecture.

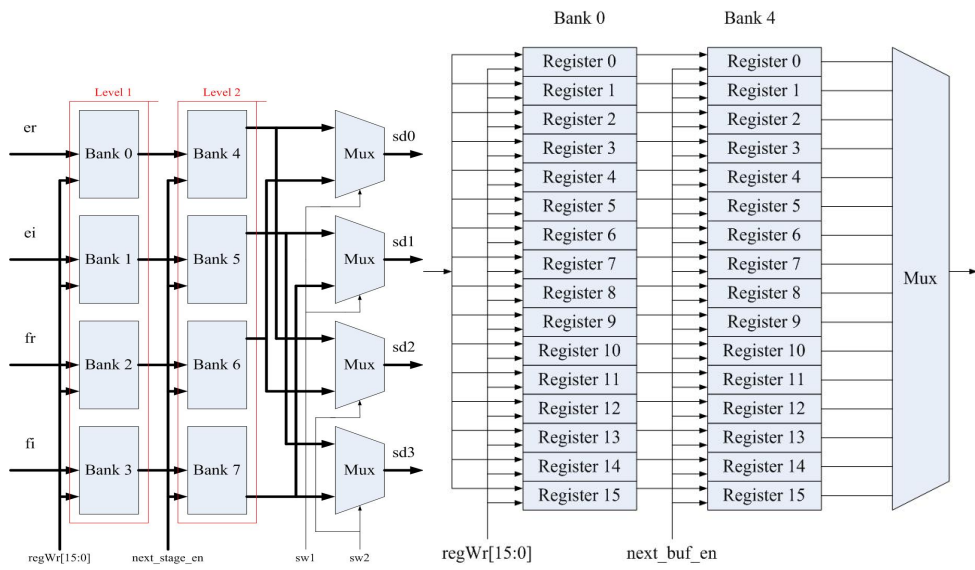


Fig. 5. The example of 32-point FFT register array architecture.

This register array adopts asynchronous design scheme to reduce the clock switching activity. The bus “regWr [15:0]” in Fig. 5 is designed by using shifter register. The initial value of this bus is 16’h0001. After 16 clock cycles, it will become 16’h8000. In addition, it becomes 16’h0000 in the stall step. In each bank, only one register would be stored and the other register would be closed. After the butterfly unit (BF) executes 16 times, the signal “next_stage_en” in Fig. 5 will be positive edge trigger and renew the data. The transformation of data address is regular in the address generator module (addr_gen). The “addr_gen” module involves two inputs (num_stage and num_buf) and two output (sw1 and sw2).

For an example of 32-point FFT, the address transform function is shown in Table 1. In this table, there are 5 stages (corresponding num_stage = 0 ~ 4) in the 32-point radix-2 FFT processor. Since each stage has to execute butterfly unit (BF) 16 times, we need 4 bits signal (num_buf) to count. After finishing calculation in stage, we will obtain 32 real

Table 1. The pattern of “addr_gen” module in a 32-point FFT processor.

num_stage = 1		num_stage = 2		num_stage = 3		num_stage = 4	
num_buf	sw2	num_buf	sw2	num_buf	sw2	num_buf	sw2
0000		0000		0000		0000	
00000	01000	00000	00100	00000	00010	00000	00001
0001		0001		0001		0001	
00001	01001	00001	00101	00001	00011	00000	10001
0010		0010		0010		0010	
00010	01010	00010	00110	10000	10010	00010	00011
0011		0011		0011		0011	
00011	01011	00011	00111	10001	10011	10010	10011
0100		0100		0100		0100	
00100	01100	10000	10100	00100	00110	00100	00101
0101		0101		0101		0101	
00101	01101	10001	10101	00101	00111	10100	10101
0110		0110		0110		0110	
00110	01110	10010	10110	10100	10110	00110	00111
0111		0111		0111		0111	
00111	01111	10011	10111	10101	10111	10110	10111
1000		1000		1000		1000	
10000	11000	01000	01100	01000	01010	01000	01001
1001		1001		1001		1001	
10001	11001	01001	01101	01001	01011	11000	11001
1010		1010		1010		1010	
10010	11010	01010	01110	11000	11010	01010	01011
1011		1011		1011		1011	
10011	11011	01011	01111	11001	11011	11010	11011
1100		1100		1100		1100	
10100	11100	11000	11100	01100	01110	01100	01101
1101		1101		1101		1101	
10101	11101	11001	11101	01101	01111	11100	11101
1110		1110		1110		1110	
10110	11110	11010	11110	11100	11110	01110	01111
1111		1111		1111		1111	
10111	11111	11011	11111	11101	11111	11110	11111

part results and 32 imaginary part results. Hence, we need 5 bits signal (sw1 & sw2) to select the register bank address. In the first stage (num_stage = 0), we store the result in the first level of the register array. When the first stage finishes calculating, the second level of the register array will be stored. In the second stage (num_stage = 1), we use signals “sw1” and “sw2” to select the result from the second level of the register array. We will transmit the input data to the butterfly unit (BF) to execute the second stage. The signals “sw1” and “sw2” are related to the signals “num_stage” and “num_buf”. We can infer the following inference:

We define two variables α and β .

$$\alpha = \text{the length of num_buf} \tag{14}$$

$$\beta = \text{num_stage} \tag{15}$$

$$\text{sw1}[\alpha - \beta] = \text{“Low(0)”} \tag{16}$$

$$\text{sw2}[\alpha - \beta] = \text{“High(1)”} \tag{17}$$

$$\text{sw1}[\alpha : \alpha - \beta + 1] = \text{sw2}[\alpha : \alpha - \beta + 1] = \text{rotating num_buf}[\alpha - 1 : \alpha - \beta] \tag{18}$$

$$\text{sw1}[\alpha - \beta - 1 : 0] = \text{sw2}[\alpha - \beta - 1 : 0] = \text{num_buf}[\alpha - \beta - 1 : 0] \tag{19}$$

For example, we refer to the Eqs. (16) and (17). In the second stage (num_stage = 1), sw1[3] is always “low” and sw2[3] is always “high”. In the third stage (num_stage = 2), sw1[2] is always “low” and sw2[2] is always “high”. In the fourth stage (num_stage = 3), sw1[1] is always “low” and sw2[1] is always “high”. In the fifth stage (num_stage = 4), sw1[0] is always “low” and sw2[0] is always “high”. The other bits can refer to Eqs. (18) and (19). For example, based on (18) when num_stage = 3 and num_buf = 4'b0111, we can find sw1[4:2] = sw2[4:2] = rotating {num_buf [3:1]} = rotating {3'b011} = 3'b101. Based on Eq. (19), we can find sw1[0] = sw2[0] = num_buf[0]. When all stages finish, the address generator will send output address. Besides, the second level will output the result one by one. Because FFT processor has three pipelines, the transfer would result in data conflict. In order to solve this hazard, we use a simple skill “stall” in Fig. 6. When each stage finishes, we only need two stalls. It can make the next stage to produce the correct data.

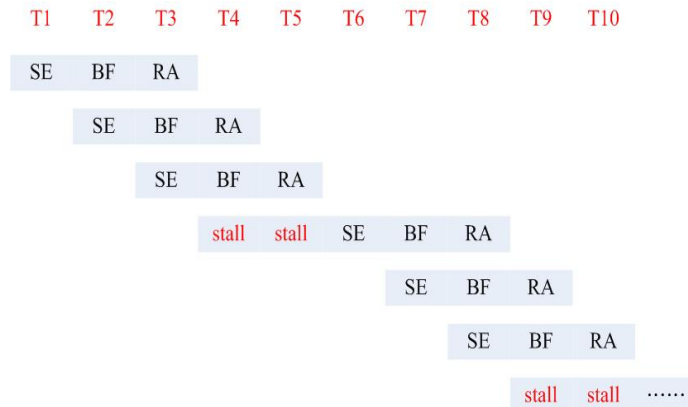


Fig. 6. The FFT processor use “stall” for solving data hazard.

Baas *et al.* [6] proposed Spiffiee FFT processor which is fabricated by full-custom design kit with $0.7\mu\text{m}$ CMOS design rule. It operates at very low supply voltages (1.1V). We use the cell-based design kit. Our chip is synthesized with TSMC $0.18\mu\text{m}$ standard cell. Although we operate at higher standard supply voltage 1.8V, we can reduce more computation cycles (clock period) in the same point FFT. This is because that the Spiffiee processor uses nine pipelines. In order to avoid the read-after-write (RAW) data hazard, Spiffiee processor needs more stalls than our FFT processor. Compared with Spiffiee processor [6], our method is very suitable for speech recognition in real time processing.

3. IMPLEMENTATION AND COMPARISON RESULTS

By previous discussion, we designed a 256-point FFT processor. It is described in synthesizable Verilog HDL. The total gate count of this FFT processor is about 136850 synthesized and estimated with TSMC $0.18\mu\text{m}$ standard library (coefficient ROM was not included). The layout view is shown in Fig. 7.

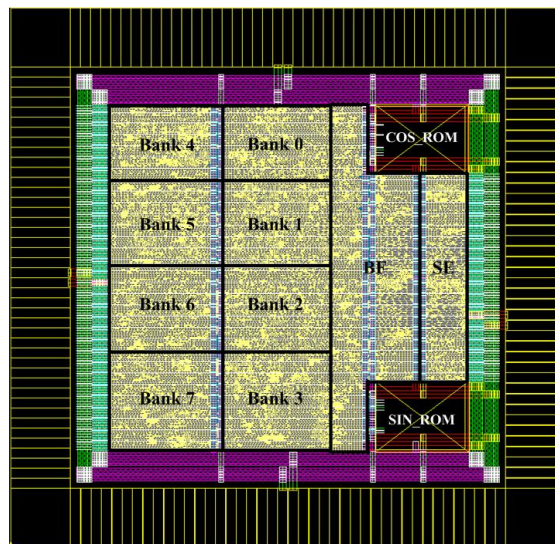


Fig. 7. The layout view of the FFT processor.

The FFT processor is available in 208-pin CQFP package. There are 167 available pins including, 8 core power pads, 24 I/O power pads, 69 input pads and 66 output pads. The maximum clock frequency of this FFT processor is about 100MHz. The format of input data is 16-bit fixed point. For the MFCC algorithm, the frame size is about 30 ms (240 samples) with a sampling frequency of 8000 Hz. For a 256-point FFT processor, the total execution time needs 1040 clock cycles. Hence, the latency is about $10.40\mu\text{s}$ ($1040 \times 10\text{ns}$). The main source of power consumption in a typical CMOS logic gate is due to the switching power, P_{sw} .

$$P_{sw} = (1/2)k C_{load} V_{dd}^2 f \quad (20)$$

where V_{dd} is the supply voltage, f is the clock frequency, C_{load} is the load capacitance of the gate, and k is the switching activity factor which is defined as the average number of times that the gate makes an active transition in a single clock cycle. In this paper, we adopt an asynchronous register array, and this issue can lower the parameter k . In addition, we also lower the load of the third pipeline because only four registers are loaded. The comparison of several recent researches is shown in Table 2. Compared with [9-12], this novel architecture can reduce more power consumption.

Finally, we use Table 3 to compare synchronous memory architecture with asynchronous memory architecture in our proposed FFT processor. Based on this table, we can see that the asynchronous circuit has smaller area. Besides, the asynchronous memory based radix-2 architecture can reduce the switching activities for the low power design.

Table 2. The comparison of several recent researches.

	Hasan[9]	Zhao[10]	Yeh[11]	Lin[12]	Proposed Low Power FFT
Technology	UMC 0.18 μ m	UMC 0.18 μ m	Avant! 0.35 μ m	0.18 μ m	TSMC 0.18 μ m
Radix	Radix-4	Radix-2	SRSDF	Radix-8	Radix-2
Power consumption (mW/MHz)	5.496	1.305	5.079	1.26	0.88
FFT point	256	256	64	8192	256
Chip-size (mm^2)	?	2.86	3.46	4.84(core)	4.73
Voltage	3.3v/1.8v	3.3v/1.8v	3.3v	3.3v/1.8v	3.3v/1.8v
Memory size	?	?	?	22K bytes	512 bytes

Table 3. To compare synchronous memory architecture with asynchronous memory architecture in our proposed FFT processor.

	Synchronous memory based	Asynchronous memory based
Technology	TSMC 0.18 μ m	TSMC 0.18 μ m
Clock rate	100MHz	100MHz
Supply voltage	3.3v/1.8v	3.3v/1.8.v
Latency	18.2 μ s	10.4 μ s
Chip area	4.91 mm^2	4.73 mm^2
Power dissipation	129.6mW	89.18mW

4. CONCLUSIONS

This paper proposed a novel FFT processor which is based on the register array. The FFT processor adopts register array to store intermediate data information instead of on-chip SRAM. This register array adopts asynchronous issue to improve clock switch-

ing activity. Compared with [9-12], this novel architecture can reduce more power consumption. This approach is very attractive for the speech feature extraction of MFCC.

REFERENCES

1. O. B. Tuzun, M. Demirekler, and K. B. Nakiboglu, "Comparison of parametric and non-parametric representations of speech for recognition," in *Proceedings of the 7th Mediterranean Electrotechnical*, Vol. 1, 1994, pp. 65-68.
2. S. S. He and M. Torkelson, "Design and implementation of a 1024-point FFT processor," in *Proceedings of the IEEE Custom Integrated Circuit*, 1998, pp. 131-134.
3. C. H. Chang, C. L. Wang, and Y. T. Chang, "Efficient VLSI architectures for fast computation of the discrete Fourier transform and its inverse," *IEEE Transactions on Signal Processing*, Vol. 48, 2000, pp. 3206-3216.
4. L. R. Rabiner and B. Gold, *Theory and Application of Digital Processing*, Prentice-Hall, Inc., 1975.
5. E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processor for VLSI implementation," *IEEE Transactions on Computers*, Vol. C-33, 1984, pp. 414-426.
6. B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE Solid-State Circuits*, Vol. 34, 1999, pp. 380-387.
7. L. H. Jia, B. X. Li, Y. H. Gao, and H. Tenhunen "Implementation of a low power 128-point FFT," in *Proceedings of the 5th International Conference on Solid-State and Integrated Circuit Technology*, 1998, pp. 369-372.
8. K. S. Stevens and B. W. Suter, "A mathematical approach to a low power FFT architecture," in *Proceedings of the International Symposium on Circuits and Systems*, Vol. 2, 1998, pp. 21-24.
9. M. Hasan, T. Arslan, and J. S. Thompson, "A delay spread based low power reconfigurable FFT processor architecture for wireless receiver," in *Proceedings of the International Symposium on System-on-Chip*, 2003, pp. 135-138.
10. Y. Zhao, A. T. Erdogan, and T. Arslan, "A low-power and domain-specific reconfigurable FFT fabric for system-on-chip applications," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005, pp. 169a.
11. W. C. Yeh and C. W. Jen, "High-speed and low-power split-radix FFT," *IEEE Transactions on Signal Processing*, Vol. 51, 2003, pp. 864-874.
12. Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A dynamic scaling FFT processor for DVB-T applications," *IEEE Journal of Solid-State Circuits*, Vol. 39, 2004, pp. 2005-2013.
13. K. S. Chong, B. H. Gwee, and S. Chang, "A low-energy asynchronous FFT/IFFT processor for hearing aid applications," in *Proceedings of the IEEE Conference on Electron Devices and Solid-State Circuits*, 2005, pp. 751-754.

Gin-Der Wu (吳俊德) received B.S. degree in Engineering Science from the National Cheng Kung University, Tainan, Taiwan, R.O.C. in 1996 and the Ph.D. degree in Electrical and Control Engineering from the National Chiao Tung University, Hsinchu, Taiwan, R.O.C. in 2000, respectively. In 2002, he joined VIA Technologies, Inc., as a

senior engineer in Chipset R&D Division. In 2003, he joined ALI Corporation, as a senior engineer in Digital AV Product Business Division. Since February 2004, he joined the National Chi-Nan University, Nantou, Taiwan, R.O.C., where he is currently an Assistant Professor of Electrical Engineering. His current research interests are speech signal processing, neural fuzzy networks, and VLSI chip design.

Ying Lei (雷穎) received the B.S. degree in Electrical Engineering from the Chung Yuan Christian University, Chungli, Taiwan, R.O.C. in 2002 and the master's degree in Electrical Engineering from the National Chi Nan University, Puli, Taiwan, R.O.C. in 2006. Then, he joined Faraday Technology Corporation, as a senior engineer in Mixed Signal Engine in 2006. His main researches are speech signal processing, DSP, and VLSI chip design.