Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering
Los Angeles, CA, USA - April 2006

Low Power Pipelined Radix-2 FFT Processor for Speech Recognition

Gin-Der Wu
Department of Electrical Engineering
National Chi Nan University
Puli, Taiwan, R.O. C.
ginderwu@ncnu.edu.tw

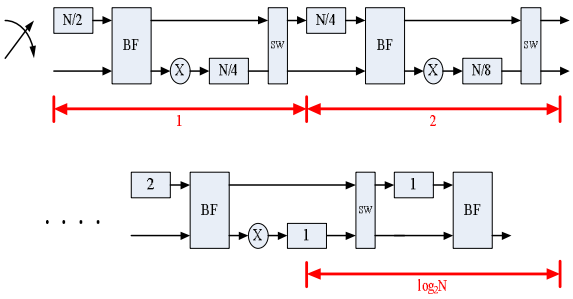
Ying Lei
Department of Electrical Engineering
National Chi Nan University
Puli, Taiwan, R.O. C.
s3323530@ncnu.edu.tw

Abstract – *It is important to develop a high-performance FFT/IFFT processor to meet the requirements of real-time, low area (low cast) ,and low power in different applications. In this paper, we proposed a radix-2 pipeline FFT processor for Mel Frequency Cepstal Coefficient (MFCC). Compared with [6]-[7], this novel architecture can reduce more power consumption. This approach is very attractive for MFCC speech feature extraction.*

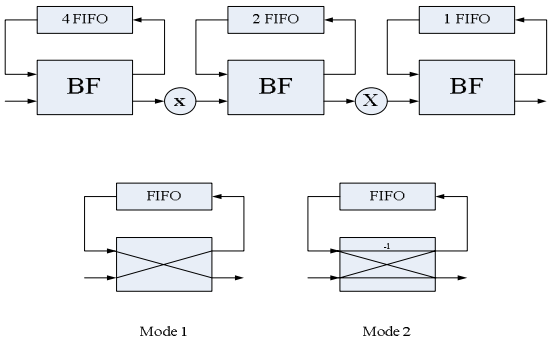
Keywords: FFT, radix-2, pipeline, Mel Frequency Cepstal Coefficient.

1 Introduction

In the field of digital signal processing, Fast Fourier Transform (FFT) plays an important role. FFT/IFFT processors are widely used in different application, such as speech process, image process, and communication system. In this paper, we focus on the field of speech process.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 Cepstal Coefficient (MFCC). Since MFCC can express human acoustic character more efficiently than LPCC, 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 in Fig. 1. , and the other is memory-based design in Fig. 2. For pipeline architecture, there are two basic hardware architectures. One is multiple-path delay commutator pipeline architecture (MDC) in Fig. 1. (a), and the other is single-path delay feedback pipeline architecture (SDF) in Fig. 1. (b). Radix-2 MDC (R2MDC) [1] 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) [2] 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 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 throughout rate.



(a)



(b)

Fig. 1. The pipeline-based FFT architecture. (a) multiple-path delay commutator pipeline architecture (MDC) (b) single-path delay feedback pipeline architecture (SDF).

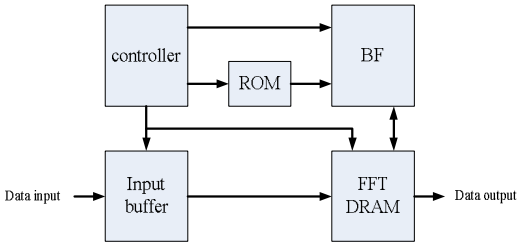


Fig. 2. The memory-based FFT architecture.

Some researchers investigated several issues of low power FFT processor. A low power cache-memory architecture was proposed in [3]. Some researchers [4] reduced power consumption by minimizing the number of complex multiplications. In [5], the authors designed a low power processor by using asynchronous elements. In [6], power saving was achieved by using the most appropriate FFT size instead of a fixed large size FFT for worst case channel conditions. In [7], 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.

In this paper, we use a butterfly unit to design our FFT processor architecture. Unlike memory-based architecture, we adopt pipeline issue to improve our hardware performance. Furthermore, we propose a novel asynchronous register array instead of memory to reduce power consumption. Generally speaking, if a digital circuit is not triggered by the clock signal, we can classify it to be an asynchronous circuit. The advantage of the asynchronous circuit is the smaller area. For example, the area of the Flip-Flop is three times large than Latch. In addition, 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. The proposed hardware architecture suits for low radix FFT processor.

2 FFT/IFFT Hardware Architecture

The flowchart of calculating MFCC is shown in Fig. 3. FFT is a very important component to calculate MFCC. In this paper, the sampling rate is 8 KHz, and the frame size is 30ms.

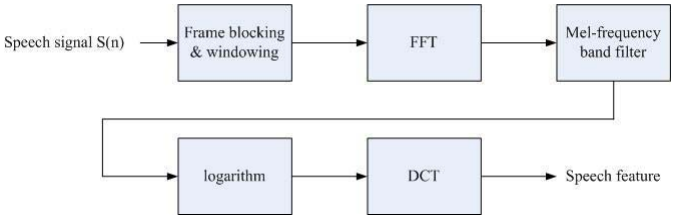


Fig. 3. the MFCC design flow.

We implement a 256-point FFT processor which has low power consumption. The proposed FFT/IFFT hardware architecture used Radix-2 Decimation-In-Frequency (DIF) algorithm. We depict even-number frequency samples as the following equations.

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

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

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

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

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

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

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

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

where r=0,1,2,…,(N/2)-1. (7)

The decomposition can be mapped to a butterfly (BF) graph which is shown in Fig. 4.

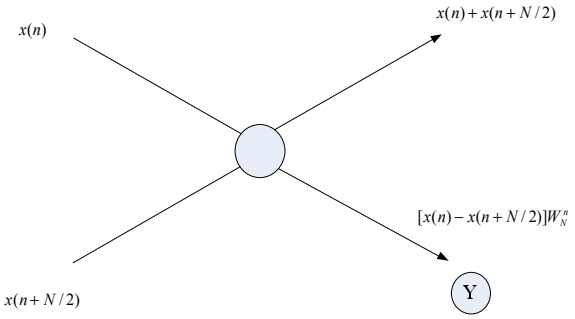


Fig. 4. The butterfly graph of the radix-2 DIF FFT.

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.

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

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

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

According to the above equations, the BF hardware architecture is shown in Fig. 5.

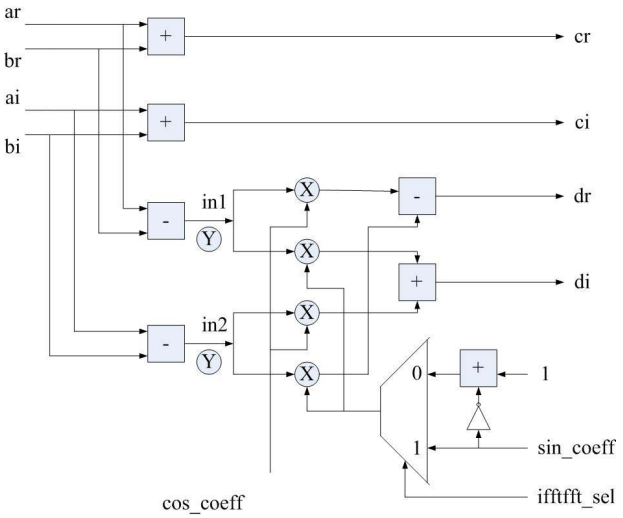


Fig. 5. The radix-2 butterfly unit (BF) architecture.

In this paper, we adopt register array to store the data instead of DRAM. As shown in Fig. 6, we use multiple-path architecture to implement the FFT processor. 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). Its architecture is based on radix-2 algorithm shown in Fig. 5. The third pipeline is register array module (RA). This module is to store the result and read the input data. In this module, we adopt asynchronous issue to reduce power consumption.

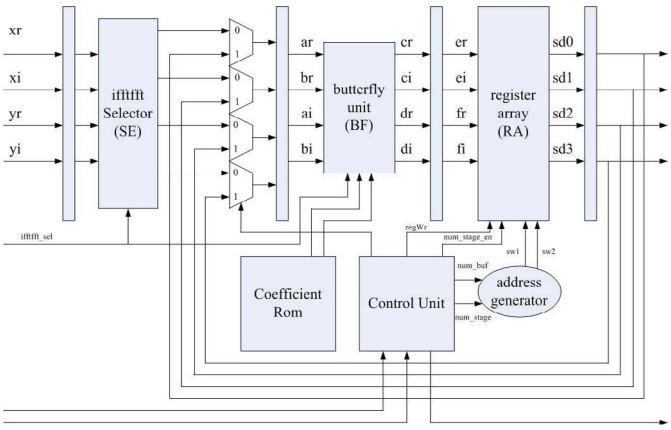


Fig. 6. The radix-2 pipeline FFT processor architecture.

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. 7. 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 ~ Bank7) would be saved when the front stage finished all execution.

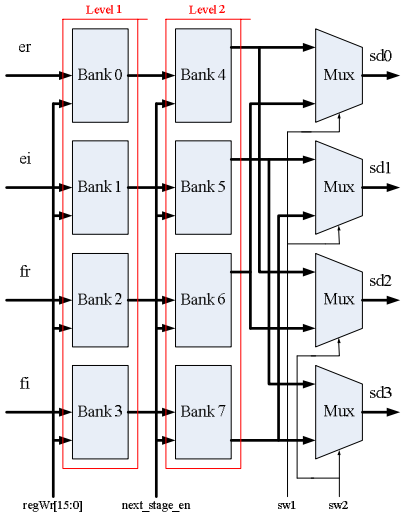


Fig. 7. The example of 32-point FFT register array architecture.

When the address generator in Fig. 6 sends the address to the register array, the second level of the register array in Fig. 7 would transmit correct data back to the BF. This register array adopts asynchronous issue to reduce the clock switching activity. The bus "regWr [15:0]" in Fig. 7 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. 7 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 (14)-(19). 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 each stage, we will obtain 32 real 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.

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

$$\beta = \text{num\_stage} \tag{15}$$

$$sw1[\alpha - \beta]=\text{"Low(0)"} \quad (16)$$

$$sw2[\alpha - \beta]=\text{"High(1)"} \quad (17)$$

$$sw1[\alpha : \alpha - \beta + 1] = sw2[\alpha : \alpha - \beta + 1]$$

$$= \text{rotating } \{ \text{num\_buf}[\alpha - 1 : \alpha - \beta] \} \quad (18)$$

$$sw1[\alpha - \beta - 1 : 0] = sw2[\alpha - \beta - 1 : 0]$$

$$= \text{num\_buf}[\alpha - \beta - 1 : 0] \quad (19)$$

For example, we refer to the equations (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 equations (18), (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 (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 8. This can ensure the next stage would produce the correct data.

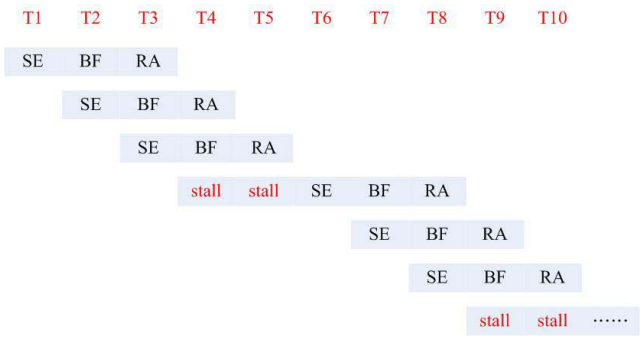


Fig. 8. The FFT processor use "stall" for solving data hazard.

3 Analysis

By previous discussion, we designed a 256-point FFT processor. It is described in synthesizable Verilog HDL. The total gate count of the this FFT processor is about 136850 synthesized and estimated with TSMC 0.18 μ standard library (coefficient ROM was not included). 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 μ s (1040×10ns).

The main source of power consumption in a typical CMOS logic gate is due to the switching power, Psw.

$$p_{sw} = (1/2)kC_{load}V_{dd}^{2}f \quad (20)$$

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 power consumption is shown in Table. 1.

Clock frequency	Hasan [6]	Zhao [7]	Proposed Low Power FFT
100M HZ	545mW (estimate)	130mW (estimate)	89.18mW
20M HZ	109.9mW	26.1mW	17.34mW

Table 1. Power consumed comparison.

4 Conclusion

This paper proposed a novel radix-2 FFT processor which is based on the register array for speech feature extraction (MFCC). This FFT processor adopt register array to store intermediate data information instead of DRAM or Dual port DRAM. This register array adopts asynchronous issue to improve clock switching activity. This novel architecture can reduce the power consumption. This architecture suits for 256 point, 128 point, and 64 point.

References

[1] L. R. Rabiner and B. Gold, "Theory and Application of Digital Processing." *Prentice-Hall, Inc*, 1975.

[2] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT Processor for VLSI Implementation," *IEEE Trans.Comput.*, C-33(5):414-426, May, 1984.

[3] M. B. Bevan, "A Low-Power, High-Performance,1024-point FFT Processor," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, pp. 380-387, March, 1999.

[4] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "Implementation of a low power 128-point FFT Processor," *in Proceedings of Fifth International conference on Solid-State and Integrated Circuit Technology*, pp. 369-372, 1998.

[5] K. S. Stevens and B. W. Suter, "A mathematical approach to a low power FFT Architecture," in *Proc. IEEE International Symposium on Circuits and Systems*, pp. II-21-II-24, 1998.

[6] M. Hasan, T. Arslan, and J. S. Thompson, "A delay spread based low power reconfigurable FFT processor architecture for wireless receiver," *IEEE International Symposium on System-on-Chip*, pp. 135-138, 2003.

[7] Yutian Zhao, Ahmet T. Erdogan, and Tughrul Arslan, "A Low-Power and Domain-Specific Reconfigurable FFT Fabric for System-on-Chip Applications," *IEEE International Symposium on Parallel and Distributed*, pp. 169a-169a, April, Nov. 2005.