A Modified Signal Flow Graph and Corresponding Conflict-Free Strategy for Memory-Based FFT Processor Design

Yinghui Tian, Yong Hei, Zhizhe Liu, Qi Shen, Zhixiong Di, and Tao Chen

***Abstract*—This brief presents a modified radix-4 fast Fourier transform (FFT) signal flow graph, whose input and output both are in natural order. Compared with the conventional radix-4 signal flow graph, it does not buffer the result of the last stage or execute the bit-reverse operation to generate the result, but generates the result directly in the last stage. Thus, the number of iterations is reduced by one. In order to realize the proposed memory-based FFT processor by using the modified radix-4 FFT signal flow graph, a conflict-free strategy and corresponding memory-addressing scheme is proposed. At last, the hardware implementation for the proposed FFT processor is proposed. Through the adoption of this method, FFT processor of arbitrary point conforming to the radix-4 algorithm can be implemented. Compared with the previous memory-based FFT processors, the proposed FFT processor has less processing time under similar or lower resource consumption.**

***Index Terms*—Fast Fourier transform (FFT), modified signal flow graph, memory-based architecture, conflict-free strategy, memory-addressing scheme.**

I. Introduction

FAST Fourier Transform (FFT) is the most important algorithm in digital signal processing area. Most of applications require FFT processor, which has the input and output both in natural order. The conventional FFT signal flow graph which has input in natural order and output in bit-reverse order is mainly applied in the FFT processor. For the conventional signal flow graph, in order to realize output in natural order, bit-reverse operation has to be executed which increases the computation time. This brief presents a modified radix-4 signal flow graph which has the input and output both in natural order. Compared with the conventional one, the modified signal flow graph avoids bit-reverse operation, thus it decreases the computation time effectively. This brief focuses on the modified radix-4 algorithm and corresponding hardware implementation.

There are two kinds of architecture to realize the FFT processor. The first one is pipeline architecture [1]-[3] which has independent butterfly unit and memory in each stage. The other one is memory-based architecture [4]-[7] which reuses the same butterfly unit and memory in each stage. This brief adopts memory-based architecture to realize the FFT processor based on the modified radix-4 signal flow graph. In order to realize memory-based FFT, solving the address conflict problem is critical. This brief also proposes a conflict-free strategy and corresponding memory-addressing scheme which is suitable for the proposed signal flow graph.

The rest of this brief is organized as follows. Section II describes the modified radix-4 signal flow graph. Section III discusses the architecture of the proposed FFT processor and the corresponding conflict-free strategy. Finally the implementation results and the comparison is provided in Section IV.

Manuscript received January 22, 2018; revised March 26, 2018; accepted April 14, 2018. Date of publication April 19, 2018; date of current version December 20, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61474135 and Grant 61504110, and in part by the Fundamental Research Funds for the Central Universities under Grant 2682015CX067. This brief was recommended by Associate Editor H. K. Lam. *(Corresponding author: Yong Hei.)*

Y. Tian is with Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China, also with School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China, and also with Beijing Institute of Remote Sensing Equipment, Beijing 100854, China (e-mail: nmgtyh@163.com).

Y. Hei is with Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China, and also with School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: heiyong@ime.ac.cn).

Z. Liu and T. Chen are with Beijing Institute of Remote Sensing Equipment, Beijing 100854, China (e-mail: liuzhizhe123@163.com; taoc90@126.com).

Q. Shen is with Beijing Research Institute, China Telecom Corporation, Beijing 102209, China (e-mail: qishen0504@163.com).

Z. Di is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China (e-mail: dizhixiong2@126.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2018.2828648

II. The Modified Radix-4 Signal Flow Graph

A. The Conventional Radix-4 Signal Flow Graph

According to the radix-4 algorithm, the conventional radix-4 signal flow graph and the twiddle factors in each stage can be obtained. Take a 16-point FFT as an example, the conventional signal flow graph is shown in Fig. 1.

B. Modified Radix-4 Signal Flow Graph

According to the fundamental theory of the signal flow graph, it is known that the signal flow graphs before and after transformation are equivalent as long as the relative position of operation of each node maintains unchanged [8]. Thus, in order to obtain the modified signal flow graph which has the property that both input and output are in natural order shown in Fig. 2, the node and corresponding twiddle factors in Fig. 1 are moved.

In Fig. 2, it is obvious that the position of the input of the butterfly unit in each stage of the modified signal flow graph is identical to the conventional one. Whereas, the position of the result of the butterfly unit in the modified signal flow graph is different from the conventional one. The four results of the

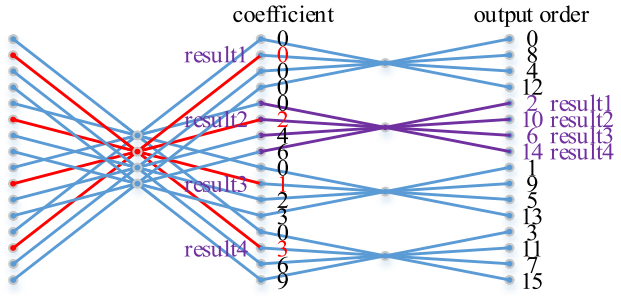

Fig. 1. Conventional radix-4 signal flow graph.

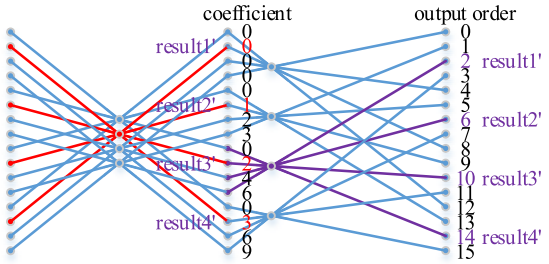

Fig. 2. Modified radix-4 signal flow graph.

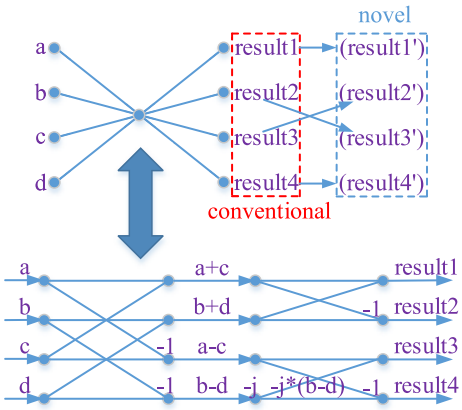

Fig. 3. Radix-4 butterfly computation.

butterfly unit in the modified signal flow graph are stored in the position of n, n+N/4, n+2*N/4 and n+3*N/4, where N stands for the number of point of FFT, n represents which butterfly unit is executed (0<n<=N/4).

In addition, the conventional radix-4 butterfly computation follows the way mentioned in the red part of Fig. 3. However, the one in the modified signal flow graph follows the way mentioned in the blue part of Fig. 3 which only exchanges the second and the third result in conventional radix-4 butterfly computation. The architecture of the radix-4 butterfly computation unit is shown in Fig. 4 which is a direct mapping of Fig. 3.

III. Proposed Architecture and Conflict-Free Strategy

A. FFT Architecture

It describes the architecture of the proposed FFT processor in Fig. 5. The architecture contains two single-port

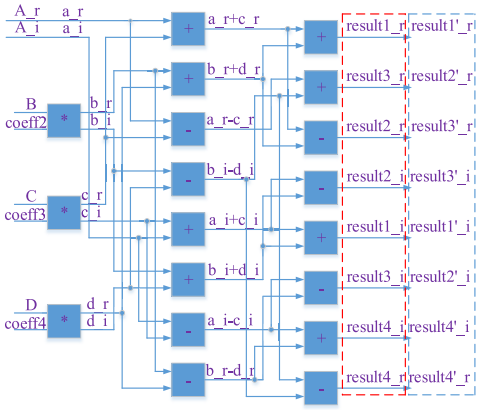

Fig. 4. The architecture of the radix-4 computation unit.

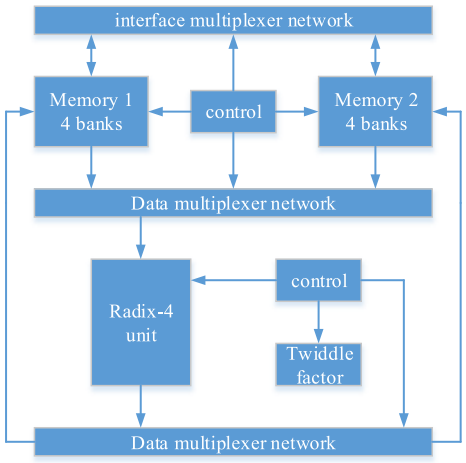

Fig. 5. The architecture of the proposed FFT processor.

memories each of which contains four banks, one radix-4 butterfly computation unit, one interface multiplexer network, two data multiplexer network, one ROM used for storing the butterfly coefficients and one controller used for controlling the state of each module in FFT processor.

The memories in proposed FFT processor are used to store the input data or the results of the butterfly unit. In addition, it generates the input data of the butterfly unit or the ultimate results of the proposed FFT processor. They work in ping-pang mode among each stage. One is used for generating four inputs of the butterfly unit simultaneously or the ultimate results of the proposed FFT processor, and the other is used for storing the input data or receiving the four results of the butterfly unit. In order to avoid the address conflict in the modified signal flow graph, the proposed conflict-free strategy is adopted. Three multiplexers in Fig. 5 select the appropriate data for the radix-4 butterfly unit and the memories.

B. Conflict-Free Strategy

In this section, the proposed conflict-free strategy is discussed in detail. This brief takes a 64-point FFT processor as an example. The conflict-free strategy is appropriate for arbitrary 4^n-point FFT processor.

1) The First Stage of FFT: At first, the input data is stored in sequence into the four banks in RAM Ping as illustrated in Fig. 6. The number in RAM Ping shown in Fig. 6 represents

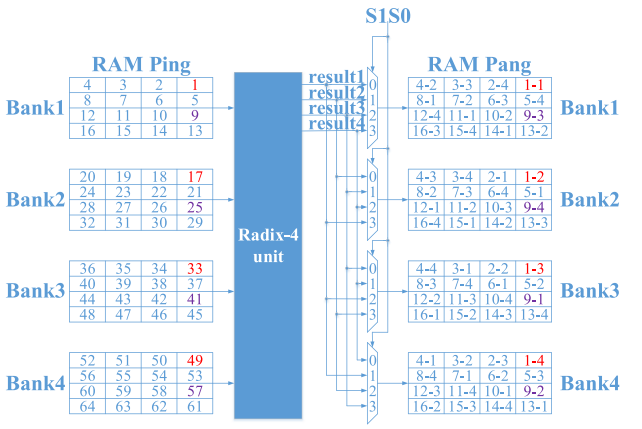

Fig. 6. The first stage of 64-point FFT computation.

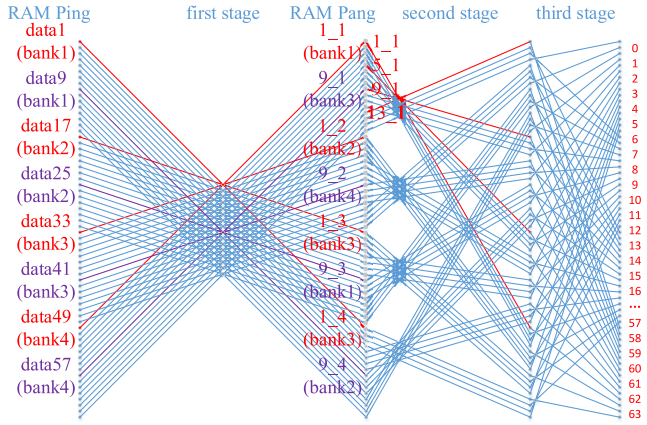

Fig. 7. Modified radix-4 signal flow graph for 64-point FFT.

the index of the input data. In Fig. 6, the address of each bank increases from right to left and from top to bottom. For example, data 17 is stored in address 1 of bank2, data 18 is stored in address 2 of bank2, data 21 is stored in address 5 of bank2 and so on.

After all the input data are stored, four data are outputted simultaneously from four banks of RAM Ping to the radix-4 butterfly computation unit according to the first stage of the 64-point modified signal flow graph shown in Fig. 7. As illustrated in the left red part of the first stage in Fig. 7 and the red part of RAM Ping in Fig. 6, data1, data17, data 33 and data 49 are outputted simultaneously at the first clock cycle from the first address of the four banks to the radix-4 butterfly computation unit. In addition, the purple part stands for the process of the ninth butterfly computation at the ninth clock cycle.

In the following computation, the address-conflict is generated because more branches of the modified signal flow graph are generated. In order to avoid this conflict in the following stages, this brief proposes a way for data storage and fetching. Table I describes the storage way for the results of butterfly unit in each stage of 64-point FFT. In Table I, the number in the first column stands for the address and computation cycle for each bank, the number in the second column stands for the value of multiplexer signal S1S0 shown in Fig. 6, and the results of the radix-4 butterfly computation in other columns are expressed as x_y, x stands for which butterfly computation

TABLE I
THE STORAGE WAY OF THE RESULT OF BUTTERFLY COMPUTATION

Address(cycle)	S0S1	Bank1	Bank2	Bank3	Bank4
4'b0000(1)	0	1_1*	1_2	1_3	1_4
4'b0001(2)	1	2_4	2_1*	2_2	2_3
4'b0010(3)	2	3_3	3_4	3_1*	3_2
4'b0011(4)	3	4_2	4_3	4_4	4_1*
4'b0100(5)	1	5_4	5_1*	5_2	5_3
4'b0101(6)	2	6_3	6_4	6_1*	6_2
4'b0110(7)	3	7_2	7_3	7_4	7_1*
4'b0111(8)	0	8_1*	8_2	8_3	8_4
4'b1000(9)	2	9_3	9_4	9_1*	9_2
4'b1001(10)	3	10_2	10_3	10_4	10_1*
4'b1010(11)	0	11_1*	11_2	11_3	11_4
4'b1011(12)	1	12_4	12_1*	12_2	12_3
4'b1100(13)	3	13_2	13_3	13_4	13_1*
4'b1101(14)	0	14_1*	14_2	14_3	14_4
4'b1110(15)	1	15_4	15_1*	15_2	15_3
4'b1111(16)	2	16_3	16_4	16_1*	16_2

and y stands for which result of the same butterfly computation. For example, as illustrated in the right red part of the first stage in Fig. 7 and the red part of RAM Pang in Fig. 6, the four butterfly computation results 1_1, 1_2, 1_3 and 1_4 are generated at the first computation cycle. And the first result 1_1 of the butterfly computation 1 are stored in the first address of bank1, and the second result 1_2 of the butterfly computation 1 are stored in the first address of bank2, and so on. The purple part stands for the storage process at the ninth computation cycle.

All storage process of other butterfly computation in the first stage is shown in RAM Pang in Fig. 6, which is consistent with Table I. The S1S0 in Fig. 6 selects one of the four results of the butterfly unit for the corresponding banks according to Table I. When S1S0=0, result1 is stored in bank1; When S1S0=1, result2 is stored in bank1 and so on. The connection relation is shown in Fig. 6.

In order to obtain S1S0, bank address $C_{n-1}C_{n-2}\ldots\ldots C_3C_2C_1C_0$ and intermediate variable $O_{n-1}O_{n-2}\ldots\ldots O_3O_2O_1O_0$ are used. For N-point FFT, the depth of each bank equals to N/4 which can be expressed by $\log_2^{N/4}$ bit binary data. Thus n equals to $\log_2^{N/4}$. From the first column of Table I, it is shown that the bank address increases in order when storing the butterfly results.

$$\begin{aligned}
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b00,\ O_{2*x-1}O_{2*x-2} = 2'b00;\\
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b01,\ O_{2*x-1}O_{2*x-2} = 2'b11;\\
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b10,\ O_{2*x-1}O_{2*x-2} = 2'b10;\\
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b11,\ O_{2*x-1}O_{2*x-2} = 2'b01.
\end{aligned}$$

Let C equals to $\sum_{x=1}^{n/2} O_{2*x-1}O_{2*x-2}$, the lowest 2 bits of C equals to S1S0. For example, when $C_3C_2C_1C_0 = 4'b0000$, C = $O_1O_0 + O_3O_2$ = 2'b00+2'b00=2'b00. Thus S1S0=2'b00.

2) The Middle Stage of FFT: For the middle stages of the FFT computation, the regulation from butterfly unit to the memory is the same as the operation mentioned in the first stage. In this subsection, more attentions are paid on the regulation from the memory to the butterfly unit.

In order to conform the second stage of the modified signal flow graph, regulation from the memory to the butterfly unit are proposed. For example, at the first clock cycle data 1_1, 5_1, 9_1 and 13_1 that conform the red part of the second

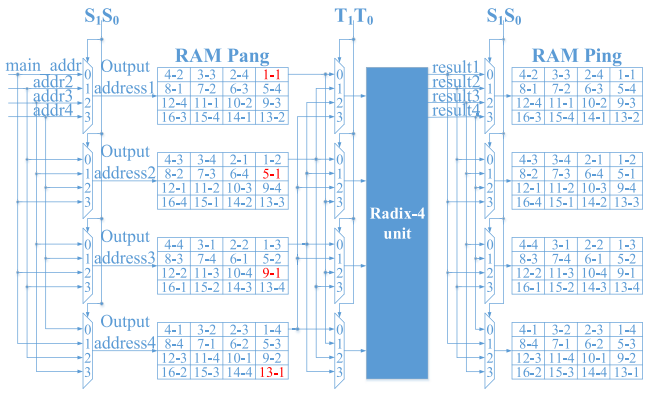

Fig. 8. The second stage of 64-point FFT computation.

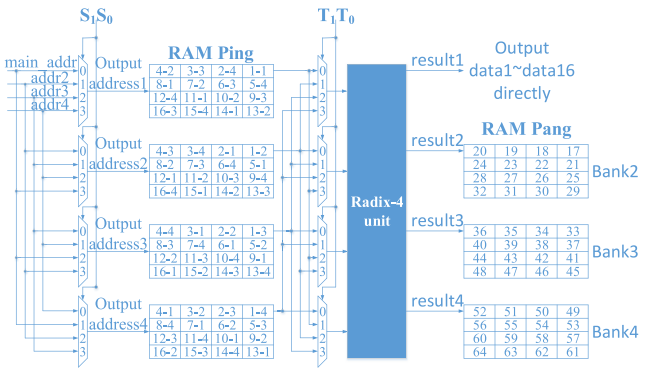

Fig. 9. The third stage of 64-point FFT computation.

TABLE II
THE OUTPUT OF MEMORY IN MIDDLE STAGE

$C_3C_2C_1C_0$ (Cycle)	$A_3A_2A_1A_0$	Bank1	Bank2	Bank3	Bank4
0000(1)	0000	1_1(0000*)	5_1(0100)	9_1(1000)	13_1(1100)
0001(2)	0001	14_1(1101)	2_1(0001*)	6_1(0101)	10_1(1001)
0010(3)	0010	11_1(1010)	15_1(1110)	3_1(0010*)	7_1(0110)
0011(4)	0011	8_1(0111)	12_1(1011)	16_1(1111)	4_1(0011*)
0100(5)	0000	13_2(1100)	1_2(0000*)	5_2(0100)	9_2(1000)
0101(6)	0001	10_2(1001)	14_2(1101)	2_2(0001*)	6_2(0101)
0110(7)	0010	7_2(0110)	11_2(1010)	15_2(1110)	3_2(0010*)
0111(8)	0011	4_2(0011*)	8_2(0111)	12_2(1011)	16_2(1111)
1000(9)	0000	9_3(1000)	13_3(1100)	1_3(0000*)	5_3(0100)
1001(10)	0001	6_3(0101)	10_3(1001)	14_3(1101)	2_3(0001*)
1010(11)	0010	3_3(0010*)	7_3(0110)	11_3(1010)	15_3(1110)
1011(12)	0011	16_3(1111)	4_3(0011*)	8_3(0111)	12_3(1011)
1100(13)	0000	5_4(0100)	9_4(1000)	13_4(1100)	1_4(0000*)
1101(14)	0001	2_4(0001*)	6_4(0101)	10_4(1001)	14_4(1101)
1110(15)	0010	15_4(1110)	3_4(0010*)	7_4(0110)	11_4(1010)
1111(16)	0011	12_4(1011)	16_4(1111)	4_4(0011*)	8_4(0111)

stage in Fig. 7 are outputted simultaneously to butterfly computation unit. As illustrated in the red part of RAM Pang in Fig. 8, 1_1 is stored in address 0 in bank1, 5_1 is stored in address 4 in bank2, 9_1 is stored in address 8 in bank3 and 13_1 is stored in address12 in bank4. Thus, four addresses of the four banks need to be generated at the first clock cycle.

According to the proposed modified signal flow graph, Table II describes the four output addresses and corresponding data of the four banks in the second stage of 64-point FFT in each cycle. The first column in Table II stands for which butterfly computation is processing, and the last four column stand for the output data and corresponding address of each bank. The address in Table II is expressed in binary, and the address labeled with "*" stands for "main address". Other three addresses of the banks can be obtained through "main address". The output process of the memory is also shown in RAM Pang of Fig. 8.

To obtain the regulation of "main address", the variable $C_{n-1}C_{n-2}\ldots\ldots C_3C_2C_1C_0$ is used to stand for the butterfly computation number. The variable $C_{n-1}C_{n-2}\ldots\ldots C_3C_2C_1C_0$ is added from 0 to maximum in sequence. For the x-th stage of N-point FFT, its "main address" $A_{n-1}A_{n-2}\ldots\ldots A_1A_0$ equals to $C_{n-1}C_{n-2}\ldots\ldots C_{n-2*x+5}C_{n-2*x+4}00C_{n-2*x+1}C_{n-2*x}\ldots\ldots C_1C_0$, where $2 <= x <= \log_4^N$. For example, when $x = 2$, $n = 4$, $A_3A_2A_1A_0 = 4'b00C_1C_0$ as illustrated in the second column of Table II. Other three addresses address2, address3 and address4 are obtained as below:

address2 $= A_{n-1}A_{n-2}\ldots\ldots A_{n-2*x+5}A_{n-2*x+4}, A_{n-2*x+3}A_{n-2*x+2} + 1, A_{n-2*x+1}A_{n-2*x}\ldots\ldots A_1A_0$,

address3 $= A_{n-1}A_{n-2}\ldots\ldots A_{n-2*x+5}A_{n-2*x+4}, A_{n-2*x+3}A_{n-2*x+2} + 2, A_{n-2*x+1}A_{n-2*x}\ldots\ldots A_1A_0$,

address4 $= A_{n-1}A_{n-2}\ldots\ldots A_{n-2*x+5}A_{n-2*x+4}, A_{n-2*x+3}A_{n-2*x+2} + 3, A_{n-2*x+1}A_{n-2*x}\ldots\ldots A_1A_0$.

For example, for the first computation cycle of the second stage in the 64-point FFT whose $n = 4$, $x = 2$, and $A_3A_2A_1A_0 = 4'b0000$, its address2=4'b0100, address3=4'b1000 and address4=4'b1100.

After the address of each of four banks is obtained, the connection relation between the four addresses and four banks needs to be established. The relationship between bank 1 and the four addresses are described emphatically which is illustrated in the left part of Fig. 8. In Fig. 8, the output address of bank1 is selected by the signal S1S0. When S1S0=0, the output address of bank1 equals to "main address"; when S1S0=1, the output address of bank1 equals to address2 and so on. Other three relationship can also be seen in the left part of Fig. 8.

In order to obtain S1S0, $C_{n-1}C_{n-2}\ldots\ldots C_1C_0$ mentioned above needs to be used.

$$\begin{aligned}
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b00,\ O_{2*x-1}O_{2*x-2} = 2'b00;\\
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b01,\ O_{2*x-1}O_{2*x-2} = 2'b11;\\
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b10,\ O_{2*x-1}O_{2*x-2} = 2'b10;\\
&\text{When } C_{2*x-1}C_{2*x-2} = 2'b11,\ O_{2*x-1}O_{2*x-2} = 2'b01.
\end{aligned}$$

And $C = \sum_{x=1}^{n/2} O_{2*x-1}O_{2*x-2}$, S1S0 equals to the lowest 2 bits of C, $0 < x <= n/2$.

3) The Last Stage of FFT: For the last stage of the FFT processor, the main distinction is the storage of the memory which is shown in the RAM Pang of Fig. 9. Just because of the property of output in nature order, the computation time and the iteration time decrease. Since the first result of the butterfly unit outputs directly as the ultimate FFT result, it does not need to store all the results of the butterfly unit into the memory and execute bit-reverse operation. In addition, when the first result is outputted the other three results of the butterfly unit are stored in bank2 to bank4. When all the butterfly computation is done, the first 16 FFT results has been outputted and the remaining 48 FFT results are stored in bank2 to bank4. After that, the remaining 48 results are read out in sequence from bank2 to bank4 as the ultimate FFT results.

TABLE III
PERFORMANCE COMPARISON

	Proposed	[9]	[10]
radix	4	4	2^3
Parallel Process	4	4	4
Memory type	single	dual	dual
Memory size	2N	N	2N
Memory bank	4	4	4
Iteration	log_2^N/2-1	log_2^N/2	-
Cycle per iteration	N/4	N/4	-
Total processing time	N(log_2^N)/8-N/4	N(log_2^N)/8	N(log_2^N)/12
Complex adder	8	8	12
Complex multiplier	3	3	4
Constant multiplier	0	0	4
Continuous flow	YES	NO	YES

IV. EXPERIMENTAL RESULT AND COMPARISON

In this section, a comparison is made in resource consumption, performance and power consumption. Since the resource consumption of the FFT processor is dominated by the memory and arithmetic logical unit (ALU), the memory size and the ALU quantity are compared emphatically. In addition, since the processing time of the FFT processor is determined by the iteration times and the cycle per iteration, the iteration times and the cycle per iteration are also compared emphatically. Through adding the registers in the critical path, the frequency of the FFT processor is enhanced. In addition, the frequency is determined by the technology used. Thus, frequency is not needed to be compared. The power consumption is related to the technology used, the resource consumption and working frequency. If all the FFT processors working under the same technology and frequency, the power is dominant by the resource consumption. Thus, the area comparison reflects the power comparison. Table III shows the comparison between the proposed FFT processor and the previous one.

Reference [9] requires a total dual port memory of size N. According to [11], the area of single-port memory is about 56% less than that of dual-port memory. Thus, the proposed one has less memory consumption compared with [9]. In addition, ALU used between proposed one and [9] is identical. Thus, the total area of proposed processor is less than [9]. Besides, the processing time of the proposed one is less than [9]. Thus the proposed FFT has a better overall result compared with [9].

Table III shows that, the processing time of [10] is N(log_2^N)/12, while the processing time of the proposed FFT processor is N(log_2^N)/8-N/4. When the number of point is less than 64, the processing time of proposed one is less compared with [10]. But when the number of point is larger than 64, the processing time of the proposed one is longer compared with [10]. Besides, [10] adopts dual-port memory whose size is 2N to realize FFT processor. According to [11] mentioned above, the memory consumption of [10] is more than twice compared with the proposed one. In addition, [10] consumes more ALU compared with the proposed one. Thus, the total area of [10] is larger than the proposed one under the same FFT size. Because the memory area is dominant when the FFT size is larger, we estimate conservatively that the area of [10] is twice as large as the proposed one. In order to compare the overall result, the product of the processing time and memory size are adopted. The product of the proposed one is 1/2*(N(log_2^N)/8-N/4) which is smaller than the product of [10] that is 1*N(log_2^N)/12. Thus, the proposed FFT processor has a better overall result compared with [10].

TABLE IV
PERFORMANCE OF THE PROPOSED FFT PROCESSOR

	N	frequency	Technology	word length	power	Area
Proposed	4096	400MHz	65nm	16+16	46.8mW	0.84mm^2

Thus, the proposed FFT processor has a better overall result compared with the previous ones. At last, Table IV shows the result of the proposed 4096-point FFT processor. The maximum frequency is 400MHz under a 65nm technology by using SYNOPSYS tool, and the power is 46.8mW under 400MHz. The area is 0.84mm^2 with 16 bit real part and 16 bit imaginary part as its input data.

V. CONCLUSION

In this brief, a modified signal flow graph for memory-based FFT processor is proposed. The proposed method reduces the number of iteration by one. We also present corresponding conflict-free strategy to realize the proposed FFT processor. After comparison, the proposed method is attractive to realize the memory-based FFT processor.

REFERENCES

[1] M. Garrido, S.-J. Huang, S.-G. Chen, and O. Gustafsson, "The serial commutator FFT," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 10, pp. 974–978, Oct. 2016.

[2] X.-B. Yin, F. Yu, and Z.-G. Ma, "Resource-efficient pipelined architectures for radix-2 real-valued FFT with real datapaths," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 8, pp. 803–807, Aug. 2016.

[3] J. Q. Liu, Q. J. Xing, X. B. Yin, X. B. Mao, and F. Yu, "Pipelined architecture for a radix-2 fast Walsh–Hadamard–Fourier transform algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 11, pp. 1083–1087, Nov. 2015.

[4] Q.-J. Xing, Z.-G. Ma, and Y.-K. Xu, "A novel conflict-free parallel memory access scheme for FFT processors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 11, pp. 1347–1351, Nov. 2017.

[5] X.-B. Mao, Z.-G. Ma, F. Yu, and Q.-J. Xing, "A continuous-flow memory-based architecture for real-valued FFT," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 11, pp. 1352–1356, Nov. 2017.

[6] Z.-G. Ma, X.-B. Yin, and F. Yu, "A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 9, pp. 876–880, Sep. 2015.

[7] Z. Qian and M. Margala, "Low-power split-radix FFT processors using radix-2 butterfly units," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 3008–3012, Sep. 2016.

[8] Y. H. Tian *et al.*, "A memory-based FFT processor using modified signal flow graph with novel conflict-free address schemes," *IEICE Electron. Exp.*, vol. 14, no. 15, pp. 1–11, Aug. 2017.

[9] M. Garrido, M. A. Sanchez, M. L. López-Vallejo, and J. Grajal, "A 4096-point radix-4 memory-based FFT using DSP slices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 375–379, Jan. 2017.

[10] K.-F. Xia, B. Wu, T. Xiong, and T.-C. Ye, "A memory-based FFT processor design with generalized efficient conflict-free address schemes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 6, pp. 1919–1929, Jun. 2017.

[11] H.-F. Luo, Y.-J. Liu, and M.-D. Shieh, "Efficient memory-addressing algorithms for FFT processor design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2162–2172, Oct. 2015.