ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.15, 1–11

A memory-based FFT processor using modified signal flow graph with novel conflict-free address schemes

Yinghui Tian[a)], **Yong Hei**[b)], **Zhizhe Liu, Zhixiong Di, Qi Shen, and Zenghui Yu**

Institute of Microelectronics of Chinese Academy of Sciences,

3 Beitucheng West Road, Chaoyang District 100029, Beijing, China

a) *nmgtyh@163.com*

b) *heiyong@ime.ac.cn*

Abstract: In this brief, we propose a novel method which realizes conflict-free strategy in memory-based FFT, of which the hardware complexity is simplified, since only a few extra registers are needed and the control logic is identical in all stages. In addition, we present a modified signal flow graph to fit for the proposed conflict-free strategy. The modified signal flow graph derives from the mixed-radix signal flow graph and has constant geometry property. Furthermore, continuous-flow is adopted to increase the throughput. Thus, the proposed FFT processor has better performance compared with the previous memory-based FFT processors. Simulation result shows that for the proposed 8 to 2048-point FFT processor, the maximum frequency is 400 MHz by using a 65-nm CMOS technology, and the area is 0.45 mm^2 in the same condition.

Keywords: fast Fourier transform (FFT), modified signal flow graph, constant geometry, conflict-free strategy, memory-based, continuous-flow

Classification: Integrated circuits

References

[1] J. Wang, *et al.*: "A mixed-decimation MDF architecture for radix-2^k parallel FFT," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **24** (2016) 67 (DOI: 10.1109/TVLSI.2015.2402207).

[2] Z. K. Wang, *et al.*: "A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 973 (DOI: 10.1109/TVLSI.2014.2319335).

[3] C. Yu and M. H. Yen: "Area-efficient 128- to 2048/1536-point pipeline FFT processor for LTE and mobile WiMAX systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 1793 (DOI: 10.1109/TVLSI.2014.2350017).

[4] X. B. Yin, *et al.*: "Resource-efficient pipelined architectures for radix-2 real-valued FFT with real datapaths," IEEE Trans. Circuits Syst. II, Exp. Briefs **63** (2016) 803 (DOI: 10.1109/TCSII.2016.2530862).

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

[5] B. G. Jo and M. H. Sunwoo: "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," IEEE Trans. Circuit Syst. I, Reg. Papers **52** (2005) 911 (DOI: 10.1109/TCSI.2005.846667).
[6] P. Y. Tsai and C. Y. Lin: "A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **19** (2011) 2290 (DOI: 10.1109/TVLSI.2010.2077314).
[7] H. F. Luo, *et al.*: "Efficient memory-addressing algorithms for FFT processor design," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 2162 (DOI: 10.1109/TVLSI.2014.2361209).
[8] Z. G. Ma, *et al.*: "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 **62** (2015) 876 (DOI: 10.1109/TCSII.2015.2435522).
[9] M. Garrido, *et al.*: "A 4096-point radix-4 memory-based FFT using DSP slices," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25** (2017) 375 (DOI: 10.1109/TVLSI.2016.2567784).
[10] K. F. Xia, *et al.*: "A memory-based FFT processor design with generalized efficient conflict-free address schemes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25** (2017) 1919 (DOI: 10.1109/TVLSI.2017.2666820).
[11] J. H. Takala, *et al.*: "Conflict-free parallel memory access scheme for FFT processors," Proc. IEEE ISCAS (2003) 524 (DOI: 10.1109/ISCAS.2003.1205957).
[12] J. Kwong and M. Goel, "A high performance split-radix FFT with constant geometry architecture," Proc. IEEE DATE (2012) 1537 (DOI: 10.1109/DATE.2012.6176717).

1 Introduction

Fast Fourier Transform (FFT) is the most important operation in digital signal processing, and is widely used in digital communication, radar system and image processing system, etc. Two kinds of commonly used FFT architectures are pipelined architecture [1, 2, 3, 4] and memory-based architecture [5, 6, 7, 8, 9]. The pipelined architecture achieves a higher throughput, but also inevitably with larger resource consumption and power consumption, since it uses independent processing element in each stage. The memory-based architecture has a lower throughput, but it has smaller resource consumption and power consumption, since it reuses the same processing elements in all stages.

In this brief, we pay more attention to the memory-based architecture due to the consideration of the area. Memory-based FFT is composed of memory, butterfly unit and corresponding control logic. The memory is used to store the input data, the temporary data and the ultimate result. Since multiple data are written into one single memory simultaneously when processing the storage operation, there exists the problem of address conflict. In order to solve the problem, conflict-free strategy is proposed.

The conflict-free strategies proposed by [8] and [10] adopt multiplexers to change the output order in each stage. The storage order in each stage is different, thus the control logic varies in each stage. To realize the reordering, extra multi-

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

plexers are needed. Hence, it adds extra multiplexers and the control logic is relatively complicated.

To optimize the conflict-free strategy further, we propose a novel method, using which the resource consumption is reduced compared with [10]. The proposed conflict-free strategy is fit for constant geometry signal flow graph in [11, 12]. Besides, the property of constant geometry signal flow graph is that all stages are identical. By using the property, the logic becomes uniform which simplify the design.

In order to improve the performance of memory-based FFT, two methods are usually used. One way is to adopt high-radix butterfly unit, and the other is to adopt several parallel butterfly units. Of course, it is a tradeoff between the resource consumption and the performance. The proposed FFT processor adopts radix-4/2 mixed-radix algorithm to realize the tradeoff. The proposed mixed-radix signal flow graph is modified to obtain the constant geometry property. By using the property of the modified mixed-radix signal flow graph, arbitrary 2^n-point FFT processor can be implemented. In order to increase the throughput, two 2048-word memories are used to implement the continuous-flow computation.

The organization of this paper is shown as follows. Section II describes the modified signal flow graph for mixed-radix algorithm. Section III describes the design issue of the proposed FFT processor. Section IV focuses on the implementation and performance evaluation. Finally, the paper is concluded in Section V.

2 Modified signal flow graph for mixed-radix algorithm

In order to realize arbitrary 2^n-point FFT with constant geometry property, the radix must equals to 2. Unfortunately, the performance is worse if the radix equals to 2. So we propose a modified radix-4/2 mixed-radix signal flow graph which has constant geometry property to get better performance. By using the proposed signal flow graph, the proposed conflict-free strategy can be obtained.

2.1 Conventional radix-4/2 mixed-radix algorithm

The Radix-4/2 mixed-radix algorithm and the radix-4 algorithm have similar structures in signal flow graph. Through combining the two algorithm, arbitrary 2^n-point FFT can be implemented.

The N-point conventional radix-4/2 mixed-radix FFT is defined as follow:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \leqq k \leqq N-1 \tag{1}$$

$$= \sum_{n_0=0}^{1} \sum_{n_1=0}^{3} \cdots \sum_{n_{L-1}=0}^{3} x(n_{L-1}, n_{L-2}, \ldots, n_1, n_0) W_N^{nk} \tag{2}$$

Where

$$nk = \left(n_0 + 2 * \sum_{i=0}^{L-2} n_{i+1} 4^i\right)\left(\sum_{i=0}^{L-1} k_i 4^i\right)$$

Where L equals to the number of stage for the radix-4/2 mixed-radix FFT algorithm

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

Where

$$n_{L-1} = 0, 1, 2, 3$$

$$n_{L-2} = 0, 1, 2, 3$$

. . . . . .

$$n_1 = 0, 1, 2, 3$$

$$n_0 = 0, 1$$

And

$$k_{L-1} = 0, 1$$

$$k_{L-2} = 0, 1, 2, 3$$

. . . . . .

$$k_0 = 0, 1, 2, 3$$

Using the formula above, the signal flow graph of the radix-4/2 FFT and the corresponding twiddle factors in each stages can be obtained. The conventional radix-4/2 mixed-radix signal flow graph is shown in Fig. 1.

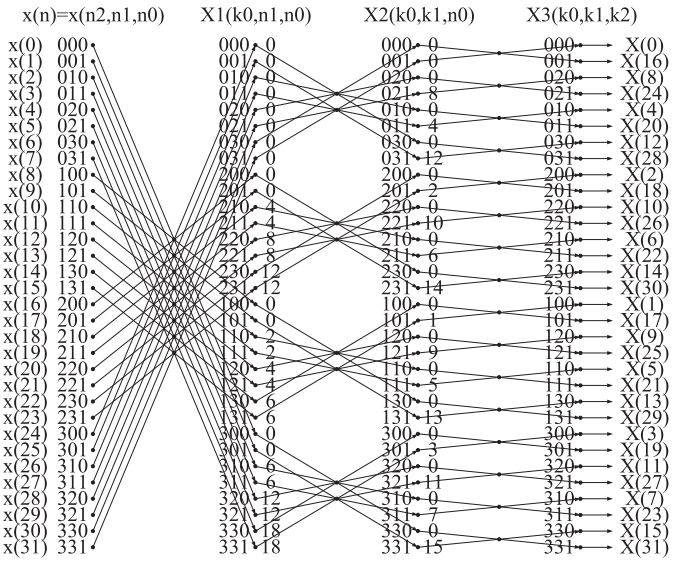

Fig. 1. Conventional radix-4/2 mixed-radix signal flow graph.

2.2 Modified signal flow graph for mixed-radix algorithm

According to the theory of signal flow graph, it is known that the signal flow graphs before and after transformation are equivalent as long as the relative operation position of each node maintains unchanged. Thus, in order to obtain the modified signal flow graph, we move the node and corresponding twiddle factors in Fig. 1. The output data in each stage are connected to the input node of the next stage in sequence. By using the strategy mentioned, the modified signal flow graph is obtained which is shown in Fig. 2.

Fig. 2 describes a 32-point FFT signal flow graph, and shows that all stages are totally identical. Using this modified signal flow graph, the proposed conflict-free strategy can be realized.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

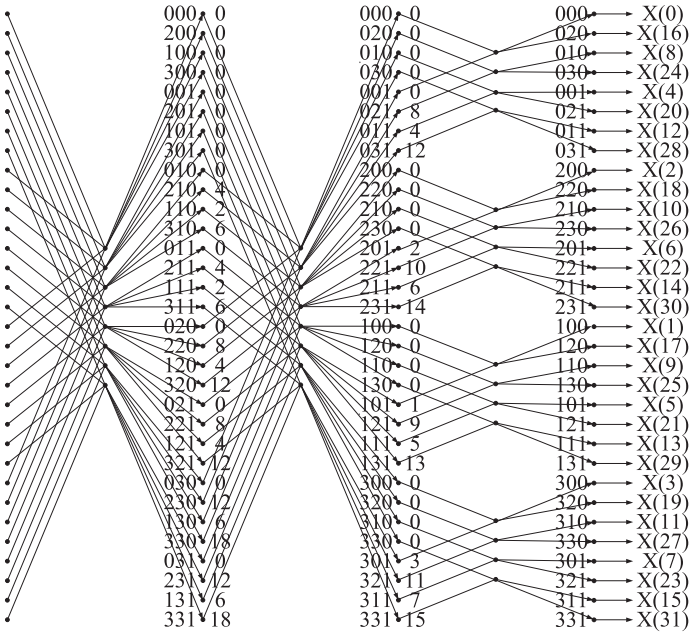

Fig. 2. Modified radix-4/2 signal flow graph.

3 Design issue of the FFT processor

3.1 Conflict-free strategy

In this section, a novel conflict-free strategy is proposed. The method solves the address conflict by using the proposed signal flow graph which is illustrated in Fig. 2. The data flow chart in Fig. 3 corresponds to the modified signal flow graph in Fig. 2. In this 32-point FFT computation, a pair of memory is used to store input data, intermediate data and ultimate result. Each memory consists of four banks. The depth of each bank equals to 2. Each address in each bank stores four data.

First, four input data are combined together one by one. Then the combined data are stored in Memory 1 in sequence. The storage sequence of the input data is shown in the first column of Fig. 3.

After all the input data are stored, four data that follows the butterfly computation positions in the first stage of Fig. 2 are read out simultaneously. The four data in Memory 1 are stored in different banks, thus the reading address conflict does not exist anymore. For example, data 1, data 9, data 17 and data 25 in Memory 1 are read out concurrently at first, and data 2, data 10, data 18 and data 26 in Memory 1 are read out concurrently next time. Because each address stores four input data, we only extract one of them each time.

After the four data in Memory 1 are read out simultaneously, the fetched data are input to the shared butterfly unit simultaneously as shown in the Butterfly Radix-4 of Fig. 3. After the butterfly computation, four results of the butterfly computation are generated. According to the input order of the second stage of the modified signal flow graph as shown in Fig. 2, four results are combined together and stored in Memory 2 in Fig. 3. For example, the butterfly results of data 1, data 9, data 17 and data 25 in Memory 1 are combined together and stored in the first address of Bank 0 in Memory 2. And the remaining storage sequence of the result of butterfly computation in Stage 1 is shown in the second column in Fig. 3.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

After all the operations above are finished, the first stage of FFT computation is completed. Because the second stage of signal flow graph is identical to the first stage of signal flow graph, the data flow chart in Stage 2 is identical to Stage 1.

The computation in Stage 3 is radix-2 computation. Thus, the data flow chart is a little bit different. According to the Stage 3 in Fig. 2, the computation sequence can be obtained. For example, the first data and the fifth data in Stage 3 operate radix-2 computation. The first data in Stage 3 which is stored in the first address in Bank 0 is data 1, as shown in the third column in Fig. 3. And the fifth data in Stage 3 which is stored in the second address in Bank 0 is data 9, as shown in the third column in Fig. 3. Data 1 and data 9 conduct radix-2 computation, and the results are stored following the sequence in the last stage of Fig. 2. Thus, they are stored in the first address of Bank 0 in Memory 2, as shown in the fourth column in Fig. 3.

In order to get better performance, two pairs of radix-2 computation are conducted simultaneously. For example, data 1 and data 9, data 3 and data 11 in column 3 are computed in parallel. The results of them are combined as one data and stored in the first address in Bank 0 in Memory 2, as shown in the fourth column in Fig. 3. The storage sequence of the result of butterfly computation in Stage 3 is shown in the fourth column in Fig. 3.

The address conflict is solved by using the proposed conflict-free strategy. The proposed method avoids the writing address conflict, since it writes the four data to one address instead of four. In addition, the proposed method avoids the reading address conflict, since it reads the four data from four different banks instead of one bank.

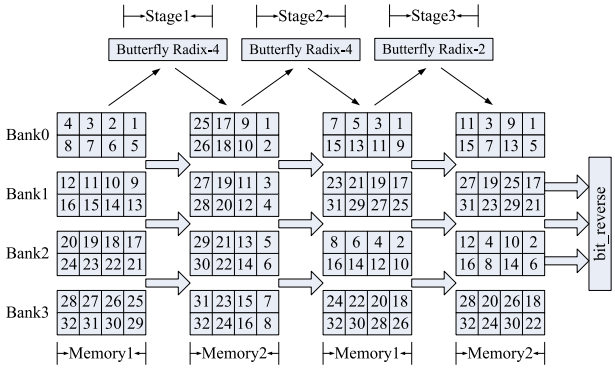

Fig. 3. Conflict-free strategy for constant geometry approach.

3.2 Proposed FFT architecture

The proposed architecture of the memory-based FFT processor based on the modified mixed-radix signal flow graph is shown in Fig. 4. By using the proposed architecture, the conflict-free strategy and the corresponding data flow chart mentioned in section A can be realized.

To achieve the continuous-flow, increase the throughput and reduce the latency, the architecture contains two memories and each memory is comprised of four banks.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

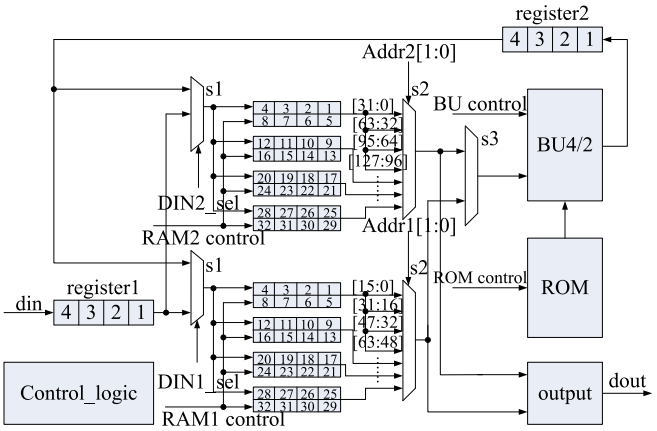

Fig. 4. The architecture for the proposed 32-point FFT processor.

The Register 1 in Fig. 4 is used as a shift register and it stores the input data one by one. After four input data are stored in the shift register, they are written into one address of the memory as a whole data.

Before radix-4 butterfly computation, four input data of the butterfly unit must be output from the four banks of one memory. Each bank outputs a whole data which contains four data, the multiplexers in the second column are used to select the definite part of the whole data as the input data of the butterfly unit (BU4/2).

The Register 2 in Fig. 4 stores the four results of the BU4/2 simultaneously. After the storage, the whole result is written into the other memory.

Before radix-2 butterfly computation, two pairs of input data of the butterfly unit must be output from one of the four banks of one memory. In order to realize computation corresponding to the last stage in Fig. 2, the two whole data which contain four data are output from the definite bank one by one and input to the butterfly unit. s2 and s3 in Fig. 4 are used to control the data path.

The control logic is used to generate the control signal and control the design to work in the desired way mentioned in Fig. 3. BU4/2 in Fig. 4 stands for butterfly unit and it can operate both radix-4 and radix-2 butterfly computation. ROM in Fig. 4 stores the twiddle factors and is also controlled by the control logic.

The more detailed structure of BU4/2 is shown Fig. 5. When conducting radix-4 computation, the four data are input directly to the basic BU4/2 module.

Whereas, when conducting radix-2 computation, the two consecutive whole data which contains four data are stored in the Register 1 and Register 2 in Fig. 5. Then two pairs of data are selected from Register 1 and Register 2 through s3 and s4 and written into the basic BU4/2 module.

The basic BU4/2 is shown in Fig. 6, and it reuses the resource of the radix-4 butterfly unit to implement two pairs of the radix-2 or radix-4 computation. The proposed basic BU4/2 is composed of three complex multipliers and eight complex adders. The multiplexers in Fig. 6 determine the data path and the radix selection concurrently.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

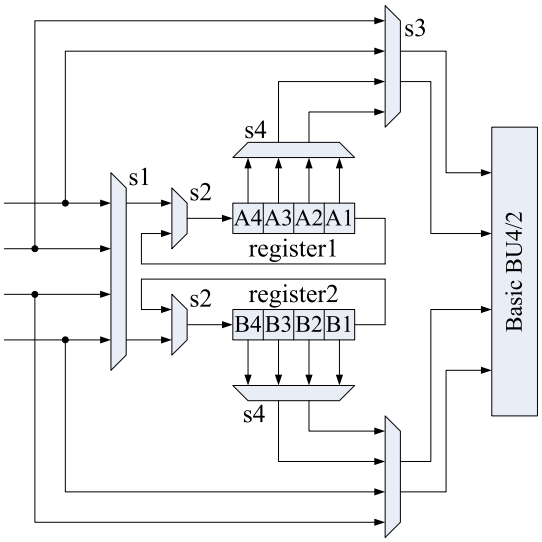

Fig. 5. The architecture of radix-4/2 butterfly unit.

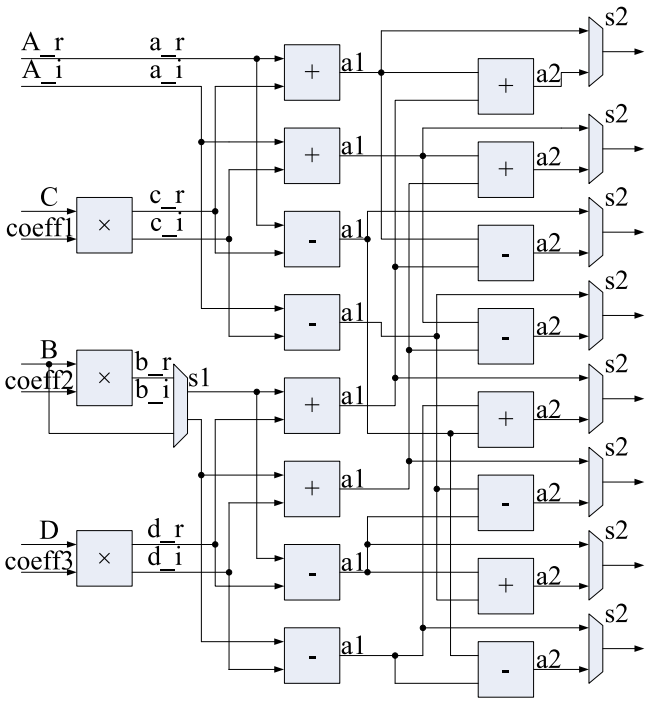

Fig. 6. The architecture of basic radix-4/2 butterfly unit.

3.3 Variable-point strategy

Based on the modified mixed-radix signal flow graph and the conflict-free strategy mentioned above, the configurable strategy can be obtained. For 8 to 2048-point configurable FFT, two groups of memories are needed. Each group of memory contains four memory banks whose depth equals to 128 and width equals to 128. When conducting configurable FFT, the maximum address in memory decreases as a result of the decrease of the FFT length. For example, when the point decreases to 1024 from 2048, the maximum address decreases to 64 from 128. Besides, the twiddle factor extraction follows the similar way.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

4 Implementation and performance evaluation

The resource consumption of memory-based FFT processor is mainly determined by the memory and the radix of the processor. To design N-point FFT processor, N or 2N word memory is needed. By using higher radix algorithm and parallel computation, the iteration cycle decreases while it consumes more arithmetic logical units. The proposed 8 to 2048-point FFT processor which adopts radix4/2 algorithm to balance the iteration cycle and resource consumption achieves a maximum frequency of 400 MHz and an area of 0.45 mm^2 by using a 65-nm CMOS technology. Table I compares various memory-based FFTs.

Table I. Comparison for previous works and this work

	proposed	[9]	[10]
radix	4/2	4	2^3
Parallel Process	4	4	4
Memory size	$2N$	N	$2N$
Memory bank	4	4	4
Iteration	$log_2^N/2$	$log_2^N/2$	-
Cycle per iteration	$N/4$	$N/4$	-
Latency of butterfly unit	4	-	0~5
Total processing time	$N(log_2^N)/8$	$N(log_2^N)/8$	$N(log_2^N)/12$
Execution time for 4096-point (μs) ($T_c + T_{proc}$)	40.96 + 61.44	81.92 + 61.44	-
Area/Normalized for 2048-point (mm^2)	0.45	-	0.86
Complex adder	8	8	12
Complex multiplier	3	3	4
Constant multiplier	0	0	4
Continuous flow	YES	NO	YES

Paper [9] in Table I adopts the conflict-free strategy which adds extra multiplexers and corresponding control logic that makes the design complicated. By using the radix-4 algorithm, [9] can only realize FFT whose length is power of four. The memory size in [9] is less compared with the proposed FFT. However, the FFT in [9] can't realize continuous flow, whereas the proposed one can realize the continuous flow so that it can increase the throughput. In order to compare the area further, arithmetic unit should also be taken into account. Table I shows that the proposed one and [9] consume the same amount of arithmetic units.

Another significant performance for the FFT processor is the execution time which is composed of total processing time and load-input-data time. In order to compare the time fairly, we extend the point number of the proposed FFT processor to 4096 by using the proposed structure. Table I shows that the proposed FFT processor and FFT processor in [9] have the same total processing time which is 61.44 μs under a working frequency of 100 MHz. On the other hand, the load-input-data time of the proposed FFT is 40.96 μs, which is half of [9]. However, considering the load frequency of the proposed FFT is twice of [9], they have the

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

same load-input-data time if under the same load frequency. Thus, they have identical execution time. Hence, the proposed FFT processor has a similar performance/resource ratio compared with [9].

Paper [10] which adopts radix-2^3 algorithm and uses a generalized conflict-free address scheme for the FFT processor does not mention the execution time of the memory-based FFT explicitly. In this paper, we proposed an equivalent way to evaluate the execution time of the proposed one and [10].

For the memory-based FFT, the execution time equals to $T_{load} + (T_i + T_b)$*Iteration, where T_{load} stands for the time of loading the input data, T_i stands for cycles per iteration and T_b stands for the latency of the butterfly unit.

For the memory-based FFT architecture, T_{load} is a constant which equals to the point number N, and T_b is a constant for specific butterfly unit. Through adding pipeline in the butterfly unit, T_b increases and the frequency of the FFT processor enhances at the same time. Compared with T_i, T_b is negligible, since the maximum T_b of the butterfly unit structure in [10] is only five, and the T_b of the proposed one is only four. Thus the execution time for the memory-based architecture approximately equals to $T_{load} + T_i$*Iteration. Considering T_{load} is a constant which equals to the point number N, the execution time is proportional to T_i*Iteration which is total processing time mentioned in Table I. According to Table I and the analysis above, the total execution time of [10] is less compared with the proposed FFT.

However, [10] consumes more resource. The memory size in [10] is the same as the proposed one, but it consumes more arithmetic units. It consumes four more complex adders, one more complex multipliers and four more constant multipliers. In order to compare the resource consumption quantitatively, we transform them under the same condition and obtain the definite area of each FFT. Paper [10] adopts a 55-nm CMOS technology and the area of [10] is 0.615 mm^2. Through the function below, we normalize the area to 65-nm technology as shown in Table I. According to Table I, the area of the proposed FFT is 52% of FFT in [10].

$$\text{Norm.Area} = \frac{Area}{(L_{min}/65\,\text{nm})^2}$$

In order to compare the overall performance with [10], we multiply the area and total processing time. The result of the multiplication reflects the overall performance of the FFT processor. The area-total-processing-time product for the proposed 2048-point FFT equals to 1267.2, while the product for [10] equals to 1614.5. Thus, the proposed FFT processor has a better performance compared with [10].

In conclusion, the way proposed in this brief is a desirable way to realize memory-based FFT processor.

5 Conclusion

In this paper, we propose an arbitrary 2^n-point FFT processor by using a modified signal flow graph and corresponding novel conflict-free strategy, so that the logic control is simplified, the hardware resource is reduced, and the frequency is higher. Thus, our proposal is very suitable for real-time and lower-resource consumption system.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017

Acknowledgments

The work is supported by National Natural Science Foundation of China (61474135; 61504110) and The Fundamental Research Funds for the Central Universities under Grant 2682015CX067.

DOI: 10.1587/elex.14.20170660
Received June 25, 2017
Accepted July 4, 2017
Publicized July 20, 2017
Copyedited August 10, 2017