International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:6, 2008

Parallel-computing approach for FFT implementation on digital signal processor (DSP)

Yi-Pin Hsu and Shin-Yu Lin

***Abstract*—** An efficient parallel form in digital signal processor can improve the algorithm performance. The butterfly structure is an important role in fast Fourier transform (FFT), because its symmetry form is suitable for hardware implementation. Although it can perform a symmetric structure, the performance will be reduced under the data-dependent flow characteristic. Even though recent research which call as novel memory reference reduction methods (NMRRM) for FFT focus on reduce memory reference in twiddle factor, the data-dependent property still exists. In this paper, we propose a parallel-computing approach for FFT implementation on digital signal processor (DSP) which is based on data-independent property and still hold the property of low-memory reference. The proposed method combines final two steps in NMRRM FFT to perform a novel data-independent structure, besides it is very suitable for multi-operation-unit digital signal processor and dual-core system. We have applied the proposed method of radix-2 FFT algorithm in low memory reference on TI TMSC320C64x DSP. Experimental results show the method can reduce 33.8% clock cycles comparing with the NMRRM FFT implementation and keep the low-memory reference property.

***Keywords*—** Parallel-computing, FFT, low-memory reference, TI DSP

I. Introduction

The signal processing plays an important role in real application, such as audio coding, image compression and video processing. Especially, data domain transformation is an essential step for above application. The discrete Fourier transform (DFT) is main and important procedure in the data analysis, system design and implementation [1]. The DFT formula can be expressed by

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nK} \quad k = 0,1,.....,N-1$$

Where $W_N^{nk} = e^{-j(2\pi/N)nk}$ is twiddle factor and x[n], X[k] are temporal and frequency signal individually. Obviously, the formula is inefficient in hardware system if programmer directly implement it.

Yi-Pin Hsu is with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan. (Corresponding author to provide phone: +886-9-39902564; e-mail: hsuyipin@gmail.com).
Shin-Yu Lin is also the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan.

In order to solve the drawback, many fast Fourier transforms (FFT) are proposed and implemented on different platforms. To reduce the complexity computation is central ideal on most of FFT algorithms. These algorithms focus on twiddle factor or radix order to perform a simply and efficient algorithm which includes the higher radix FFT [2], the mixed-radix FFT [3], the prime-factor FFT [4], the recursive FFT [5] and low-memory reference FFT [6]. In general, one is application-specific integrated circuits (ASIC) system such as [7-8]. The ASIC-based system can fit real application for low-power or high performance; however, it is very hard to modify the function. Thus the flexibility is not enough. Furthermore another is digital signal processing (DSP) system such as [9] which can achieve wide-range design by software. Although DSP-based system also keeps a high enough performance, the result is lower than ASIC-based system. Today, commercial product has a lot of considerations such as cost, performance, flexibility, and convenience implementation. The DSP-based system becomes a favorable solution.

Nevertheless, an efficient FFT algorithm be implemented on DSP is very difficult. Although TI [10] proposes an efficient FFT algorithm for C64x DSP system, data-dependent condition leads to increase clock cycle. In [6], in order to avoid multi time swap in the same data, a reorder FFT structure is proved. In Fig. 1, a 16-pts radix-2 decimation-in-frequency (DIF) FFT structure, each twiddle factor is fetched only once. For example, the twiddle factor of W_{16}^0 in the step 4 is selected to execute butterfly; but W_{16}^0 disappear in the other step. Hence, memory reduction method would reduce total memory usage and decrease power consumption as well.

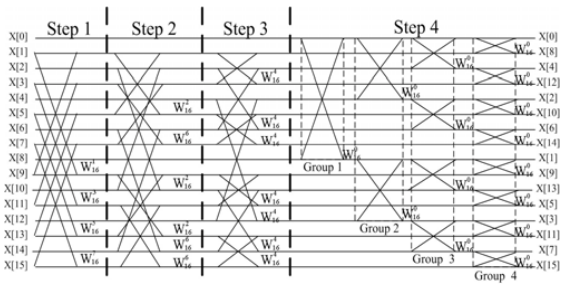

Fig. 1. A structure of NMRRM in 16-pt radix-2 DIF FFT diagram

By careful observation, the final step (i.e. step 4) has a serious problem for parallel execution in Fig. 1. First, the step 4 can divides into four groups. Because between each part has data-dependent relation, a series of flow is performed. Even if NMRRM FFT algorithm can improve the data usage, the problem of series process is still existence. The drawback leads to increase the clock cycle and waste hardware resource if the DSP has numerous operation units. Hence, to develop a parallel flow and hold the low-memory reference property algorithm becomes an important issue.

In this paper, we propose a novel algorithm to make parallel-computing flow from NMRRM FFT algorithm. Based on hardware representation, we first build two blocks and allocate individual operation unit. Afterward, we modify the data flow in the final two steps to perform a parallel-processing flow. In implementation part, the TI TMSC320C64x DSP is used as verification target. Experimental results demonstrate that the parallel-processing flow can dramatically reduce clock cycles.

The rest of this paper is organized as follows. In section II, we will brief introduce NMRRM FFT algorithm. In section III, a parallel-processing flow is designed. The experimental results are shown on section IV. Finally, some useful conclusions are demonstrated.

II. Review of Novel Memory Reduction Methods

In this section, we will brief introduce NMRRM FFT structure which is based on TI's library. Afterward we will show the implementation of radix-2 DIF FFT from NMRRM FFT as a main template for FFT algorithm on DSP.

In embedded system, the memory access is a key point on performance expression. A large number of clock cycles will be increased if memory swap is frequently executed. In Fig. 1, NMRRM FFT algorithm is proposed in order to reduce the memory swap, shows a simple and an efficient architecture. We can clear observe that each twiddle factor is used only once on each step; it is no reference in the other steps. The data-independent property is main ideas to perform this structure. For example, the calculation of twiddle factor W_{16}^{0} in the step 1, 2 and 3 can be moved to the step 4 and no influence in the result. The same skill also is applied to calculate the twiddle factor of W_{16}^{4} which is grouped into the step 3. Although this skill reduces the memory reference, the data-dependent effect is performed in the final step and can not run on pipeline structure. For example, in Fig. 1, each group from 1 to 4 has mutual dependence property. The part 2 can not be executed if part 1 runs processing. For this reason, NMRRM FFT algorithm has data-dependent status.

III. A Parallel-Computing FFT Approach on DSP

The traditional FFT has a parallel structure for DSP processing, but its drawback is multi-time reference in the same twiddle factor. The revised FFT, NMRRM FFT, can successfully avoid above fault but it performs a data-dependent problem in final step. In order to keep the advantage and discard the disadvantage between both two different structures, we propose a parallel-computing approach on FFT for hardware implementation. The twiddle factor will be taken only once and parallel-processing in each step. In order to analyze the data flow, in Fig. 2, we redraw NMRRM FFT structure which is based on 16-pt radix-2 FFT algorithm.

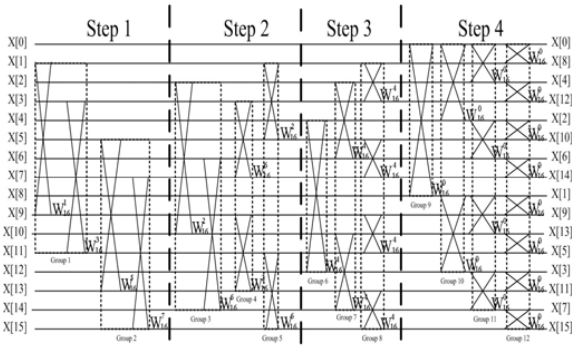

Fig. 2. Partitioning of NMRRM in16-pt radix-2 DIF FFT diagram

A. Algorithm analysis

Based on Fig. 2, from group 9 to group 12 in the step 4, each group result will affect the next and later group. For example, we can not skip the group 10 and direct to compute the group 11. Moreover, in the step 3, all groups are independently on computing process. If we change group 6 and group 8, the result is unchangeable. In further, three pairs of the group 6 with group 10, group 7 with group 11 and group 8 with group 12 has relation respectively. For example, the group 8 will be computed in first if we want to execute the group 12. Based on this constriction, thus, we can technologically combine step 3 and step 4 into a new step. Due to the step 3 and step 4 only use one twiddle factor respectively, the both twiddle factors are different, the new step needs two twiddle factors. Under the new group, the memory reference and usage equals the steps 3 and step 4 in NMRRM algorithm.

B. Hardware analysis

Each group has even element of twiddle factor from the group 1 to group 12 except group 6 and group 9 in Fig. 2. Furthermore, the numbers of twiddle factor are unbalanced in each group. The structure is inefficiently to directly implement on DSP system. Especially, the redundant hardware resource will be exhibited if the DSP system has numerous operation units. For example, we assume that a processor has eight operation units which can simultaneously run per clock cycle including four additions and subtractions; however, only one addition and one subtraction are required in the group 6. Thus the remainder operation units will be wasted. In future, the DSP system needs four clock cycles to compute eight twiddle factors in the group 12. Even if dual core processor is used, the hardware resource of processor still locates on idle status when

sequence algorithm is running.

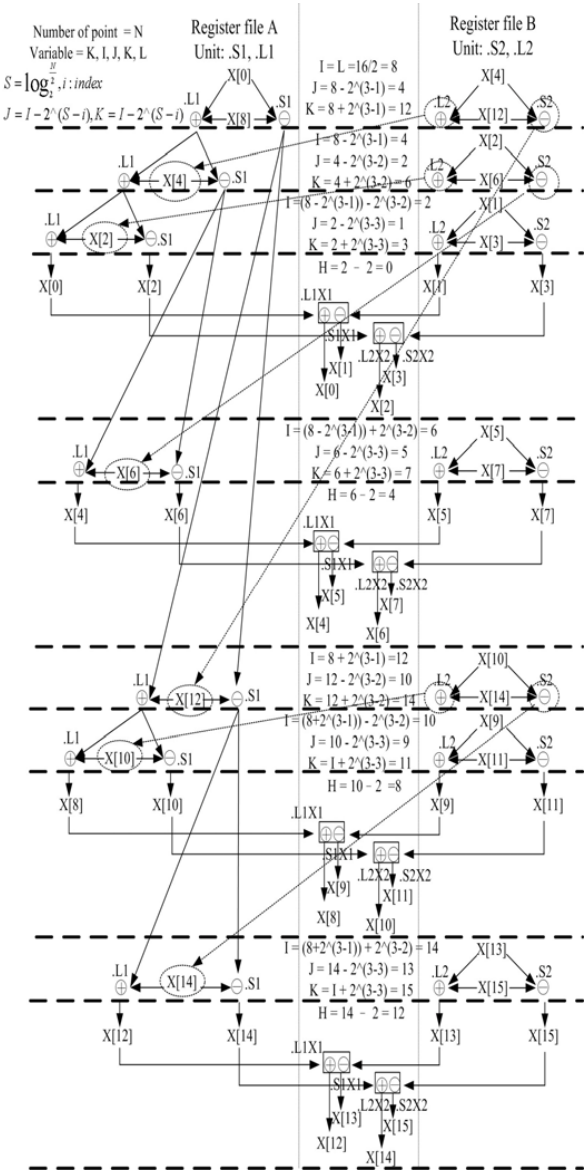

Fig. 3. Parallel-computing approach after combine the step 3 and 4 in 16-pt radix-2 DIF NMRRM FFT diagram

C. The rule of new algorithm

From above analysis, based on computational order, we can arbitrarily move the twiddle factor to perform a new structure which includes different steps and groups. In addition, only final two steps need to modify. Because the step 1 and step 2 has even number of twiddle factor in group for symmetric parallel design. Thus we propose a parallel-processing flow which depends on NMRRM FFT algorithm, take 16-pt FFT as example, and it can be separated into two main flows in Fig. 3. One is in left part and the variable I, L are its data index; another use J and K as its variable in the right part. In vertical direction, the main goal is to explain that the computation unit is independent and how many units are needed. Each stage provides output for next is also independent in the horizontal direction. The starting points are x[0] and x[L], by pre-calculate the index and following the formula set:

$$\begin{cases} I = L = N/2 \\ S = \log_2^{(N/2)} \\ J = I - 2\hat{}(S - i) \\ K = I + 2\hat{}(S + i) \end{cases}$$

Where N is N points in FFT and i is program index.

It can provide for left and right parts to compute butterflies firstly. This output is prepared for butterflies in the next stage. Following on this order, the left and right part can simultaneously process butterfly when these index are pre-decided.

In order to improve processing speed, most of DSP has numerous operation units. For example, in Fig. 4, TI TMSC320C64x [12] series includes two register files which has eight parallel units totally. However, the operation of addition and subtraction only are supported by four units (includes .S1, .L1, .S2, .L2). The data flow will occurs cross status when butterfly is executed. For example, in Fig. 3, the right part usually performs output for next calculation in left part. Based on two register file or dual core architecture, the data exchange becomes an essential issue. The data can be changed between file A and file B using 1X and 2X operator units. The syntax can be expressed as ".M2X" in assembly code which means data is processed in register file A by M unit and passed into some register in register file B by 2X. Thus the data can be arbitrarily exchanged between two register file and the system only delays one cycle in order to keep the transfer data safety. The redundant unit (such as M. and D.) are allocated to perform data address and some mathematical application. Beside, the core provides automatically 16K byte in level 1 as cache memory for data operation and 256K byte for programmable cache/RAM in level 2. Although level 2 can provides large enough memory, the 1024-pts FFT based on our new algorithm only use level 1 cache to allocate essential memory. Thus an efficient approach to allocate unit becomes an important work.

Due to the butterfly needs one addition and subtraction in radix-2, it will use two units. Namely, the maximum usage of butterfly is bounded on two units. In Fig. 3, we apply our approach in the DSP system and take 16-pt FFT as example. It can be clear obtained that two parallel-processing flows are executed in two register files, and no delay slot occurs. The unit of .L1 and .L2 can operate addition instruction in respective register file, as well as the units S1 and .S2 will operate subtraction instruction. In detail, we treat each dotted line in

horizontal direction as independent stage. It means that every stage will be finished per cycle. After, these units are uniformly separated to perform a balanced computation.

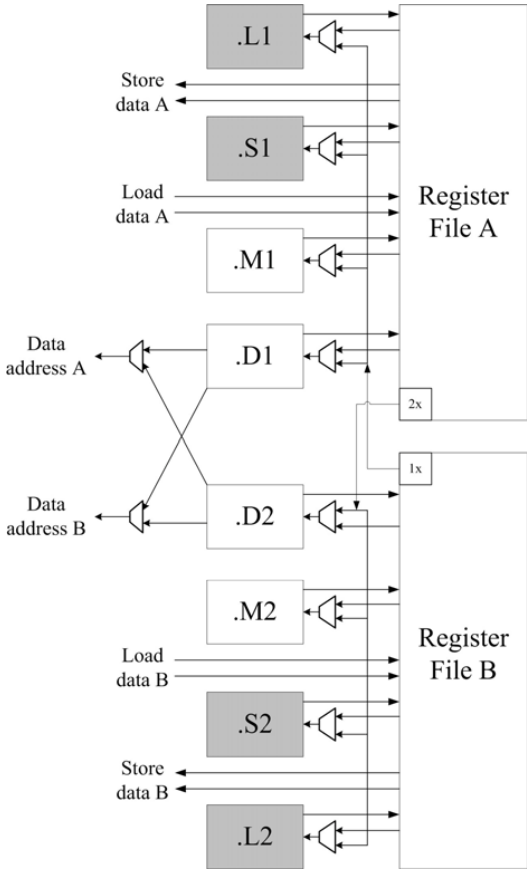

Fig. 4. The core architecture of TMSC320C64x

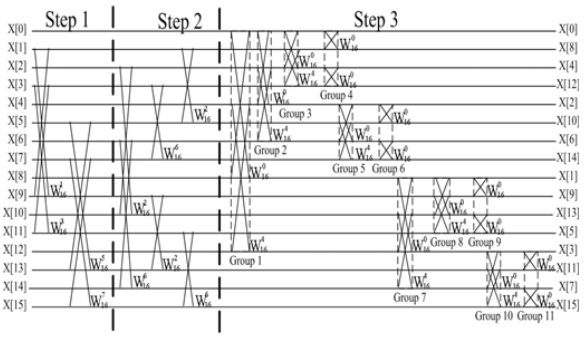

Fig. 5. Integrating NMRRM and our approach in 16-pt radix-2 DIF FFT diagram

A parallel-computing DSP-based DIF FFT algorithm is evidently depicted in Fig. 3. Besides, a parallel form is integrated in Fig. 5 which is a 16-pt DIF FFT structure. The step 3 and step 4 are combined into one step, namely new step 3, and the stage 3 is split as group from 1 to 11. In further, the step 1 and step 2 has equal properties which also can be separated into the same mode. Each group includes two twiddle factors which are mutual independence. Based on this reason, the parallel-processing can be also realized in dual-core system.

TABLE I
Comparison of Reduction Clock Cycles in radix-2 DIF FFT

FFT size	16	32	64	128
NMRRM (C1)	929	1929	3929	7929
Our approach (C2)	574	1222	2515	5164
Reduction ratio ((C1-C2)/C1)×100%	38.21%	36.65%	35.99%	34.87%

TABLE II
Comparison of Reduction Clock Cycles in radix-2 DIF FFT

FFT size	256	512	1024
NMRRM (C1)	15929	31929	63929
Our approach (C2)	10640	22150	46736
Reduction ratio ((C1-C2)/C1)×100%	33.20%	30.63%	26.89%

IV. Experimental Results

We have applied the parallel-computing approach to implement the radix-2 DIF FFT on TI TMSC320C64x DSP, which is a fixed-point property and based on very long instruction word (VLIW) architecture. The comparison item only includes number of clock cycles, in table I and table II with different FFT size, due to the memory reference is equal and NMRRM FFT has better performance than TI's library. The clock cycles are measured by TI's development environment calls as code composer studio (CCS), and current version is CCS v3.1 which also provides a suitable interface and build-in library for programmer debug such as print function. For example, data stream input and output are important skill if user wants to check relative results. Beside, the JTAG has high data rate and convenient properties are used as interface between PC and target. In order to give a fair comparison between both, all effective function in compiler option will be kept on original setting. The measurement function calls as "profile" which is plug-in and accurate in CCS, the item of clock of profile will be enabled to exactly measure the clock cycles.

The experimental results show that our approach has lower clock cycles than NMRRM FFT in radix-2 DIF FFT and average of 33.8% reduction in the number of clock cycles, in addition, the approach also keep low-memory reference property.

V. Conclusions

In this paper, a parallel-computing FFT approach on DSP is proposed. The approach is based on low-memory reference property to perform two parallel flows. The performance still equals to NMRRM FFT on radix-2 model which has better performance than TI's library. TI TMSC320C64x DSP is taken as verification system which has multiple multiply-accumulate

units is very suitable for our algorithm. The experimental results demonstrate that our approach can efficiently reduce 33.8% clock cycles and hold the low-memory reference property. In future, due to two parallel flows, our approach also can be applied on dual-core system. Thus no any bottleneck is considered when the algorithm applies on DSP-based embedded system.

REFERENCES

[1] A. V. Oppenheim and C. M. Rader, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ:Prentice-Hall, 1990, 0137549202.
[2] G. D. Bergland, "A radix-eight fast-Fourier transform subroutine for real-valued series," *IEEE Trans. Audio Electroacoust.*, vol. AE-17, no. 2, pp.138-144, Jun. 1969.
[3] R. C. Singleton, "An algorithm for computing the mixed radix fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. AE-17, no. 2, pp.93-103, Jun. 1969.
[4] D. P. Kolba and T. W. Parks, "A prime factor FFT algorithm using high-speed convolution," *IEEE Trans Acoust. Speech, Signal Process.*, vol. ASSP-25, no. 4, pp.281-294, Aug. 1977.
[5] A. R. Varkonyi-Koczy, "A recursive fast Fourier transform algorithm," *IEEE Trans Circuits Syst. II*, vol. 42, no. 9, pp.614-616, Sep. 1995.
[6] Y. Wang, Y. Tang, Y. Jiang, J. G. Chung, S. S. Song and M. S. Lim, "Novel memory reference reduction methods for FFT implementation on DSP processors," *IEEE Trans Signal Processing*, vol. 55, no. 5, pp.2338-2349, May 2007.
[7] Y. Zhou, J. M. Noras and S. J. Shephend, "Novel design of multiplier-less FFT processors," *signal processing*, vol.87, Issue 6, pp. 1402-1407,June 2007.
[8] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J . Solid-State Circuits*, vol. 34, no. 3, pp.380-387, Mar. 1999.
[9] L. He and X. Liao, "Specialising for High Performance FFT Algorithms Based on Fixed-Point DSP," *in Proc. IEEE Int. Conf. Communication, Circuits and Systems*, Vol. 1, pp. 563-566, June 2006.
[10] "TMS320C64 DSP Library Programmer's Reference (Rev. G)," Texas Instrument, Oct., 2003, SPRU7198G.
[11] "TMS320C6000 Programmer's Guide," Texas Instrument, Mar., 2006, SPRU1981.
[12] "TMS320C64/C64x+ DSP CPU and Instruction Set Reference Guide," Texas Instrument, Aug., 2006, SPRU732C.

Yi-Pin Hsu was born in Taitung, Taiwan, in 1981. He received the B.S degrees in electrical engineering from the Private Chinese Culture University (PCCU), Taiwan, in 2003 and the M.S. degrees in Department of Mechanical Electrical in 2005 in National Taiwan Normal University (NTNU). He is now a Ph.D. candidate in electrical and control engineering at National Chiao-Tung University (NCTU), Taiwan. His research interests are in the areas of parallel computing, image signal processing and DSP-based embedded system.