Transpose-free variable-size FFT accelerator based on-chip SRAM

Lei Guo, Yuhua Tang, Yuanwu Lei, Yong Dou, and Jie Zhou

National Laboratory for Parallel and Distributed Processing,
National University of Defense Technology, Changsha 410073, China
a) yuanwulei@nudt.edu.cn

Abstract: This paper presents a transpose-free variable-size fast Fourier transform (FFT) accelerator on a digital signal processing (DSP) chip. Several parallel schemes are utilized to calculate a batch of small-size FFT algorithms to achieve high performance and throughput. For middle- and large-size of FFT, we propose a transpose-free Cooley-Tukey scheme that uses the random access feature of on-chip SRAM memory to avoid the DDR access of matrix with column-wise and improves the utilization of DDR bandwidth. Experimental results show that our FFT accelerator, implemented with 65 nm library and run at 500 MHz, can achieve the energy efficiency improvement by two orders of magnitude compared with Intel Xeon CPU and obtain above 50× performance improvement compared with TI TMS320C64X DSP chip.

Keywords: fast fourier transform (FFT), accelerator, Cooley-Tukey scheme, DDR memory, SRAM

Classification: Integrated circuits

References

[1] A. Pedram, J. McCalpin and A. Gerstlauer: ASAP (2013) 175. DOI:10.1109/ASAP.2013.6567572
[2] D. H. Bailey: ACM/IEEE Conference on Supercomputing (1989) 234. DOI:10.1145/76263.76288
[3] J. R. Geraci and S. M. Sacco: ACM Transactions on Architecture and Code Optimization (TACO) (2012) 23. DOI:10.1145/235585.2355596
[4] Y. Dou, J. Zhou, Y. Lei and X. Zhou: ASAP (2007) 95. DOI:10.1109/ASAP.2007.4429964
[5] Y. Wang, Y. Tang, Y. Jiang, J. Chung, S. Song and M. Lim: IEEE Trans. Signal Process. 55 (2007) 2338. DOI:10.1109/TSP.2007.892722

1 Introduction

Fast Fourier Transform (FFT) is a fundamental tool in the domain of digital signal processing (DSP) and is widely used in digital communication, sensor signal processing, and synthetic aperture radar (SAR). FFT size varies greatly very much for different applications; for example, FFT size is smaller than the
magnitude of $K$-point in digital or wireless communication, but reaches the magnitude of $M$-point in for SAR application. Therefore, it is basic requirement of DSP applications to provide high performance and efficiency for variable-size FFT.

However, the performance and efficiency of FFT algorithms vary dramatically with different platforms and implementation methods. The computation complexity of FFT is $O(N \cdot \log^2 N)$, where $N$ refers to FFT size. The ratio of computation and communication is smaller than that of matrix operations, such as matrix multiplication and decomposition. Thus, the performance of FFT algorithms is typically limited by memory bandwidth and efficiency of data motion rather than by computation capability.

FFT algorithms typically perform poorly on general-purpose platforms, because the power-of-two strides of the FFT algorithm interact poorly with set-associative cache, set-associative address translation mechanism, and power-of-banked memory subsystem [1]. FFTW, developed by M. Frigo et al., is known as the fastest software implementation of the FFT algorithm. Corresponding to different architectures and memory-access patterns of general-purpose platform, FFTW uses the Cooley-Tukey FFT algorithm to decompose the large FFTs into small FFTs [2]; thus, it works best on array size with small prime factors with powers of two being optimal and large prime being worst case. However, FFTW does not draw out the performance of general-purpose processor; for example, the efficiency of 1 million point FFT on an IBM QS20 is only 9.3% [3].

The alternate way to implement FFT algorithm is to design special-purpose FFT hardware in FPGA or ASIC platform to achieve high performance and energy efficiency. Many related works construct fixed-point FFT hardware (typically 16- and 32-bit) for small FFTs in DSP applications, such as 32-bit FFT accelerator for 8- to 1024-point FFTs in TI C55XX serial DSP chips. However, the number of works related to floating-point FFT hardware is considerably less, especially for large FFT implementations. Pedram et al. [1] proposed a hybrid FFT/Linear Algebra core to efficiently perform FFT algorithm. Moreover, the memory hierarchy and data access patterns of FFT have been explored to calculate large one-dimensional (1-D) FFT with Cooley-Tukey scheme. However, Pedram et al. just analysed the on-chip memory hierarchy, the capability of which is just the magnitude of M bytes. Therefore, FFT size is limited by the capability of on-chip memory.

This paper explores the hardware structure on DSP chip for variable-size FFT algorithms. To improve the performance and efficiency, several parallel schemes are presented. We then propose a transpose-free Cooley-Tukey scheme for middle- and large-size of FFTs. This scheme uses the random access feature of (Static Random Access Memory) SRAM memory on DSP chip to avoid the DDR access of matrix with column-wise and improve the utilization of DDR bandwidth.
2 Background

2.1 Cooley-Tukey scheme for large 1-D FFTs

The Cooley-Tukey scheme is the most commonly used method to decompose a large 1-D FFT into many smaller FFTs, which can be calculated in parallel. Let \( N = N_1 \times N_2 \) be large-size FFT, and \( N_1 \) and \( N_2 \) be the values of power-of-two: 

\[
X[k_1N_2 + k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_2N_1 + n_1] e^{-j \frac{2\pi n_1 k_1}{N_1}} e^{-j \frac{2\pi n_2 k_2}{N_2}},
\]

where \( 0 \leq k_1 < N_1 \) and \( 0 \leq k_2 < N_2 \). The steps in the Cooley-Tukey scheme are as follows:

- **Step 1.** Calculation of column-FFT: \( N_2 \) simultaneous \( N_1 \) point FFTs are calculated as 
  \[
  \sum_{n_2} x[n_2N_1 + n_1] e^{-j \frac{2\pi n_2 k_2}{N_2}}.
  \]

- **Step 2.** Calculation of compensated twiddle factor: The result of step 1 is multiplied by 
  \( e^{-j \frac{2\pi n_1 k_1}{N_1}} \).

- **Step 3.** Calculation of row-FFT: \( N_1 \) simultaneous \( N_2 \) point FFTs are obtained as 
  \[
  \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_2N_1 + n_1] e^{-j \frac{2\pi n_1 k_1}{N_1}} e^{-j \frac{2\pi n_2 k_2}{N_2}}.
  \]

The access patterns of data matrix are as follows: the matrix is read row-wise twice and column-wise once and then written row-wise once and column-wise twice, as shown in Fig. 1(A). These access patterns significantly affect the utilization of memory bandwidth [4].

2.2 Characteristics of DRAM and SRAM

DRAM chips, such as DDR2 and DDR3, are used to store the initial data, middle results and final translated results for large FFT algorithm. DRAM bandwidth utilization varies for different access modes because of the row-buffer structure of DRAM and the burst-oriented access patterns. In the row-buffer structure, several burst commands in the same row can be executed serially to access data after the data in the row are stored in row buffer. Several cycles are added to access two elements in different physical row; these cycles used to close one physical row (writing the data in row buffer into a physical row) and open the other physical row (read the data in physical row into row buffer). In the burst-oriented access patterns, DDR2 and DDR3 must prefetch four and eight contiguous data, respectively, from the row buffer per memory access. Thus, the access of row-wise-ordered contiguous data utilizes considerable DDR bandwidth. However, DDR performance greatly diminishes when the column-wise-ordered random data are accessed. In the current paper, \( T_b \) and \( T_s \) refer to the cycles for each burst access and row change.

By contrast, on-chip SRAM memory features with high-speed and flexible access. Therefore, it is the main medium used to build high-speed share memory among multiple cores in CPU and DSP. The capability of on-chip SRAM has increased with the development of technology; for example, 4 MB on-chip SRAM in TI C66XX serial DSP chips.

Our previous work in [4] proposed an array address translation algorithm and customed special-purpose SDRAM controller, which mapped a logic window matrix into a physical row of SDRAM to increase the practical I/O
bandwidth through the balance speed of row-wise and column-wise matrix access. However, this method is rarely used with general-purpose DDR controller in DSP chips. In the current paper, we propose a transpose-free FFT accelerator on DSP chip, which uses the random access feature of on-chip SRAM to avoid column-wise matrix access and obtain high performance.

3 Transpose-free Cooley-Tukey scheme for large FFT based-on SRAM

This section presents the organization of IEEE-754 single-precision floating-point FFT accelerators on DSP chip and the transpose-free Cooley-Tukey scheme for large FFT based-on on-chip SRAM memory.

In this paper, the SRAM capability is $T \text{ byte}$ and $N_m = T/8$. $N_s$ is the maximum FFT size, which can be directly implemented using the FFT accelerator, and $N_b = N_s \times N_s$. $N_s$, $N_m$, and $N_b$ are the values of power-of-two. We use the following definitions:

- **Small-size FFT**: the FFT size is not greater than $N_s$.
- **Middle-size FFT**: Size ($N$) satisfies the condition that $N_s < N \leq N_m$.
- **Big-size FFT**: Size ($N$) satisfies the condition that $N_m < N \leq N_b$.
3.1 FFT accelerator on DSP chip

The FFT accelerator mainly consists of a FFT total controller, DMA bus controller, and FFT computation array, as depicted in Fig. 2(A).

The FFT total controller controls the run of the FFT accelerator. This module decomposes a batch of FFT operations into several sub-operation sequences, such as SRAM-read, SRAM-write, DDR-read, DDR-write, and FFT-calculation. These sub-operations overlap to enhance performance. The FFT total controller also automatically implements the Cooley-Tukey scheme for middle- and large-size FFTs.

The DMA bus controller is responsible for data motion among SRAM, DDR, and FFT computation array. This module translates the sub-operations of SRAM-read, SRAM-write, DDR-read, and DDR-write into a series of bus access transactions. These transactions access on-chip SRAM either in sequence or with a fixed stride, and access off-chip DDR memory with burst access mode.

The FFT computation array module is composed of two FFT-PE modules and a CORDIC compensated twiddle factor generator module. The number of compensated twiddle factors, determined in step 2 of Cooley-Tukey scheme, corresponds to the FFT size (N) and the storage capability of these factors is 8N bytes. Compensated twiddle factors are dynamically generated using a full-pipelined CORDIC processor instead of factor read-only memory in our design; thus, less hardware logic is used to support variable-size FFT. Each FFT-PE consists of an FFT computation controller, two butterfly modules, multi-bank data memory with a “ping-pong” structure, and twiddle factor memory, as shown in Fig. 2(B). This module can independently implement a small-size FFT.
3.2 Transpose-free Cooley-Tukey scheme

3.2.1 Parallel schemes for a batch of small-size FFTs

A batch of small-size FFT algorithms is evaluated by several parallel computation schemes, as the space-time diagram shown in Fig. 3. (1) First, a full-pipeline butterfly module is customized to exploit the pipeline parallel for small-size FFTs. This module is composed of four IEEE-754 single-precision floating-point multiplications and six floating-point additions, (2) Second, two butterfly modules and multi-bank data memory is presented to evaluate small-size FFT in parallel. Parallel schemes (1) and (2) are used to reduce the FFT computation time of each FFT-PE. (3) Third, two FFT-PE modules evaluate two different small-size FFTs in task-level parallel. This parallel scheme reduces the number of small-size FFT for each FFT-PE. (4) Finally, “ping-pong” structure of data memory is presented to overlap the overhead of data motion between SRAM (or DDR) and FFT accelerator and the overhead of FFT computation. Thus, the overhead of initial data read and result write are almost hidden by the overhead of FFT computation, as shown in Fig. 3.

3.2.2 Transpose-free Cooley-Tukey scheme for middle-size FFT

For middle-size FFT ($N = N_1 \times N_2 \leq N_m$), the capability for initial data, middle result and final result is $8N$ bytes and $8N \leq 8N_m = T$; therefore, these data can be fully stored in on-chip SRAM. As shown in Fig. 1(A) and Fig. 1(B), compared with original Cooley-Tukey scheme, transpose-free Cooley-Tukey scheme for middle-size FFT adds two steps (Step 1 and Step 4) to avoid DDR access with column-wise. In Step 1, the initial data are read from DDR memory in sequence and similarly written to SRAM. In Step 4, the final result matrix ($N_1 \times N_2$) are read from SRAM with column-wise and written DDR memory in sequence. About $2N$ cycles, instead of three time DDR access with column-wise, are needed for Step 1 and Step 4.

Therefore, the Cooley-Tukey FFT scheme is just run on SRAM with random access mode, as Step 2 and Step 3 shown in Fig. 1(B). The calculation of compensated twiddle factor (Step 2-2) is integrated into the calculation of column-FFT (Step 2-1). The column-FFT and row-FFT are treats as batch of small-size FFTs; thus, the parallel schemes are used to reduce the time of middle-size FFT.
### 3.2.3 Transpose-free Cooley-Tukey scheme for large-size FFT

For large-size FFT \( N = N_1 \times N_2 \leq N_b \), the capability of on-chip SRAM is not large enough to store the entirety of initial data, middle result, and final result. So, a block transform method is proposed for the transpose-free Cooley-Tukey scheme of large-size FFT. The capability of each block is equal to that of SRAM.

As shown in Fig. 1(C), the whole initial data matrix is initially divided into \( m = 8N/T \) column-wise blocks, called column-block. Each block is treated as a \( N_1 \times t_1 \) matrix, where \( t_1 = T/(8N_1) \). For each column-block, column-FFT and compensated twiddle factor (Steps 1, 2-1, 2-2, and 3) are executed in serial. In Steps 1 and 3, the column-block moves between DDR memory and SRAM. For DDR, we access \( t_1 \) data through burst access mode, and then switch to the next row. Thus, the cycles for Steps 1 and 3 are approximately \( 2(T/8 + N_1 \times T_1) \).

The result matrix of compensated twiddle factor is divided into \( m \) row-wise blocks, called row-block. Each block is treated as a \( t_2 \times N_2 \) matrix, where \( t_2 = T/(8N_2) \). For each row-block, row-FFT (Steps 4-1, 4-2, and 5) are performed in serial. The \( t_2 \times N_2 \) times \( N_2 \)-point FFT are calculated with the parallel scheme and the results are buffered at SRAM. In Step 5, the results of data in row-FFT are written to DDR memory with burst access mode, and then switch to the next row. Therefore, the cycles of this step are approximately \( (T/8 + N_2 \times T_2) \).

Thus, compared with original Cooley-Tukey scheme, transpose-free Cooley-Tukey scheme for large-size FFT adds three steps (Step 1, 4, and 5) to avoid DDR access with column-wise. The cycles of these steps are approximately \( (T/4 + N_1 \times T + N_2 \times T_s) \) and are smaller than \( 3T \times (T_d + T_s)/8 \), which are the overhead to access this block thrice with column-wise.

### 4 Experiments

#### 4.1 Prototype

We built an FFT accelerator in our home-made DSP chip, which has an on-chip SRAM 1MB and runs at 500 MHz. The bandwidths of DDR bus and SRAM bus are 8 GB/s (128-bit * 500 MHz). The maximum FFT size to be directly implemented with FFT accelerator is 1K, namely \( T = 1 \text{ MB}, N_s = 1 \text{ K}, N_m = 128 \text{ K}, \) and \( N_b = 1 \text{ M} \). The data memory in FFT-PE is organized into eight “ping-pong” sub-bank structure. Each sub-bank is a 256 * 64-bit RAM with two read/write ports. Thus, this data memory can provide enough ports for two butterfly modules.

Our FFT accelerator is implemented with 65 nm technology library in worst case condition. The delay of key path is 1.29 ns and total power consumption is 362 mW. The accelerator can run successfully at a clock frequency of 500 MHz. The total area of FFT accelerator is 1.9 mm², and the parallel computation array module takes 94% of the area. The overhead for other logic accounts for the remaining 6%.
Table I. Performance comparison among different platforms

| Platform     | FFT Size | 64   | 256  | 1024 | 64 K  | 1 M  |
|--------------|----------|------|------|------|-------|------|
| Intel CPU    | Time (us)| 0.171| 0.642| 3.158| 593.48| 31074|
|              | Performance (Gflops) | 11.23 | 15.95 | 16.21 | 8.83 | 3.37 |
| TI DSP       | Cycles   | 3142 | 16044| 80592|       |      |
|              | Time (us) | 0.141| 0.577| 2.808| 532.66| 17790|
|              | Cycles   | 71   | 288  | 1403 | 266328| 8894996|
| Our Accelerator | Performance (Gflops) | 13.6 | 17.75 | 18.23 | 9.84 | 5.89 |
|              | Speedup (vs. Intel) | 1.21 | 1.11 | 1.12 | 1.11 | 1.75 |
|              | Speedup (vs.TI) | 48   | 55.7 | 57.4 |       |      |

4.2 Performance comparison

Table I compares the performance of our FFT accelerator with Intel Xeon CPU X5675 (3.07 GHz) and TI TMS320C64X DSP chips. The fastest FFT library, FFTW-3.3-DDL64, is utilized to measure the performance of our design. This library was developed by M. Frigo et al. and features an automatic vectorizer to generate SSE4 SIMD instructions for Intel processors. According to these measurements, we achieve speedup by a factor of \( \frac{1.1}{C} \) to \( \frac{1.8}{C} \). However, we can achieve improvement on energy efficiency by above two orders of magnitude (36.1 GFLOPS/W vs. 0.12 GFLOPS/W); this enhancement is attributed to the specialized structure for a batch of small-size FFTs and the organization of SRAM and DDR memory for large-size FFTs. For TI TMS320C64X DSP chips, the optimized radix-2 DIF FFT codes, developed by Y. Wang [5], are compared in Table I, and we can achieve a 53\( \times \) average performance improvement.

5 Conclusion

This study explores the highly effective and energy-efficient structure of an FFT accelerator on a DSP. In this structure, several schemes are proposed to exploit the pipeline-level, instruction-level parallel and task-level parallel for a batch of FFTs. Due to the row-buffer structure and burst-oriented access patterns of DDR chip, the practical DDR bandwidth is very low for random data access with column-wise order. Thus, we present a transpose-free Cooley-Tukey scheme for large-size FFT, which uses the random access feature of on-chip SRAM to avoid the DDR access of matrix with column-wise. Experimental results show that our FFT accelerator achieve high performance and energy efficient.

Acknowledgments

This work is supported by NSFC 61125201.