A Data-Flow Approach To Accelerate Real-Valued Fast Fourier Transform

Amin Sahebi*,‡,1, Lorenzo Verdoscia†2, Roberto Giorgi†,3

* Università degli Studi di Siena, Siena, Italy
† Institute for High Performance Computing and Networking, CNR, Naples, Italy
‡ Università degli Studi di Firenze, Firenze, Italy

ABSTRACT
This paper presents a new methodology to exploit the benefit of using twiddle factor symmetry feature in a Data-Flow model of real-valued fast fourier transform —RFFT. Since there are many other libraries and benchmarks specifically working on FFT, it’s worthwhile to investigate the new algorithm in this area. We show in this paper a mindset of efficient FFT according to rearranging and due to interesting features of complex numbers to diminish the output delay time for 8-point FFT. In particular, we show using data-flow specification in a FFT application can lead a better execution time in comparison to the famous currently used libraries and algorithms.

KEYWORDS: FFT Algorithm, Data-Flow Program Graph, Real-Valued FFT

1 Methodology and background

A Fast Fourier Transform (FFT), introduced by J. W. Cooley and J. W. Tukey in the mid 1960s [CT65], known as Decimation-in-time (DIT) and later Discrete Fourier Transform (DFT). In theory, by assuming \( x_n \) as a complex N-point finite sequence value for input signal, the DFT is defined as:

\[
X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N - 1
\]

Where, \( e^{-j\frac{2\pi}{N}kn} \) can be replaced by \( W_k^N \) as called "twiddle factor", referred to the root-of-unity complex multiplicative constants which rotates in increments in order to number of samples, N [MGS66].

According to Eq. 1 a direct computation of an N-point DFT needs \( O(N^2) \) complex multiplications and addition/subtraction numbers. Moreover, as the property of in-place memory addressing, this algorithm used by many current FFT works [CQYC05]. Briefly, many traditional algorithms which are based on Cooley-Tukey algorithm, are computing even-indexed

---

1 E-mail: sahebi@diism.unisi.it
2 E-mail: lorenzo.verdoscia@cnr.it
3 E-mail: giorgi@dii.unisi.it
Figure 1: Butterfly demonstration for traditional DFT

and odd-indexed according to Eq.2 and then integrating them to the output by considering that twiddle factors features such as, Periodicity, Symmetry and Recursion [LV19].

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{kn}$$

By considering the DFT of Even-indexed of $x_{2n}$ by $E_K$ and the DFT of Odd-indexed $x_{2n+1}$ by $O_K$. Therefore we can rewrite the Eq.3 as called butterfly demonstration in this context as shown in Fig.1

$$X_k = E_k + W_k^N O_k$$
$$X_{k+N/2} = E_k - W_k^N O_k$$

Previous studies fully covered the butterfly radix-r (e.g radix-2, 4) and the principals of the FFT about the mathematical backgrounds and also hardware implementation. Moreover, to achieve the purpose to reduce the number of this operations there are good studies in Eq.2 such as [GPG09, Duh86, JF07, SJHB87, GC14, CQYC05].

In the next section we describe why we chose Data-Flow for FFT calculation and we propose a new approach with take advantage of reducing operations based on arithmetic studies. The organization rest of this study describe our proposed method, then the evaluation comparison in the experimental setup and compare it to other well-known libraries finally we speak about conclusion and our future works.

## 2 Methodology

In Data-Flow architecture, as shown in Fig.2 the structure of the system, here FFT algorithm, is seen as a series of transportation of sequentially sets of input data, where data and operations are independent of each other. So in this context we exploit this feature when data enters in to the system and then flows through the algorithm procedure one at a time they are assigned to the determined destination. According to this fact, in the Fig.3 we proposed our approach for a DFT 8-point algorithm. In detail, if investigate more in the algorithm can be observed that it considers the bit-reversal function for a specific input data, then in the line 6 and 7, the effect of using this feature that multiplication by $(-j)$ can be replaced by exchanging the real and Imaginary parts and just modifying the signs. The implementation of this algorithms are deposited in a general repository [RFF]. and we did not apply symmetry and redundancy aspect of the behavior to calculating the 8-point FFT output. In Fig.4 Pseudo algorithm [SG08] is a depth-first recursive radix-2 DIT Cooley-Tukey FFT to compute a DFT of a power-of-two size $n = 2^m$. In addition we have to consider that this algorithm is an out-of-place inputs and produces in-order output data, so in this algorithm
which is considered as FFTW algorithm plans does not include a separate bit-reverse operations [SG08], indeed, FFTW use a planner to adapt its algorithms based on the hardware configuration, so FFTW in this step returns a “reasonable” plan, and as they mentioned is not necessarily the fastest way [F05]. A plan is an executable data structure that accepts input data and computes the desired DFT.

![Figure 2: A Data-Flow scheme of the FFT algorithm](image)

![Figure 3: RFFT algorithm based on [LV19]](image)

### 3 Experimental Comparisons

According to [LV19], the experimental setup for the proposed 8-point modified RFFT was 3.6 GHz AMD Ryzen 7 1800x with GCC compiler Version 4.4.7. The time measurement and accuracy which performed to acquire this results are based on using gettimeofday resolution, although already there are some cycle counters which can be configured due to the operating system, it should be our future works to implement a precise hardware counters to measure the correct time of execution however here the situation for all algorithms are the same.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Compiler</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecFFT</td>
<td>GCC</td>
<td>348</td>
</tr>
<tr>
<td>FFTW</td>
<td>GCC</td>
<td>248</td>
</tr>
<tr>
<td>Real Value FFT</td>
<td>GCC</td>
<td>160</td>
</tr>
</tbody>
</table>

### 4 Conclusion and future works

We present a new Data-Flow approach to present a methodology for FFT algorithm to accelerate the output time delay besides reducing the number of operations in compared to
other traditional algorithms. However the main goal of this approach is to provide a Data-Flow program graph and generalize it in future works for any point FFT algorithm hope to diminish the delay output time in comparison with other popular libraries such as FFTW. In this work we consider the whole DPG of the 8-point FFT algorithm in the algorithm and time measurement. Based on our experiment results for 8-point FFT real-value input data we achieved 1.5x faster to reach the outputs.

References


