Vectorizable Design and Implementation of FFT Based on Fused Multiply-add Architectures

Junyang Zhang, Yang Guo & Xiao Hu*
College of Computer, National University of Defense Technology, Changsha, Hunan, China

ABSTRACT: This paper proposed a high efficient method by using fused multiply-add instruction to map FFT algorithms based on vector processors. According to the architecture feature of YHFT-Matrix, combing shuffle needs with memory access requests to reduce shuffling pattern, and also the method which utilizes software pipelining to fully exploit instruction-level and data-level parallelism of FFT algorithms. Then the calculating performance is improved. Experimental results show that FFT algorithms achieve high computing performance and speedups. For instance, after adopting FMA instruction optimization, the chip’s computational efficiency of 1024-point double-precision floating-point FFT algorithm is about 10% higher than before.

Keywords: FMA; FFT; vector processor; software pipeline

1 INTRODUCTION

FFT as a basic tool for time domain and frequency domain transformation, is the modern signal processing systems, including radar, sonar, seismic signal analysis, spectrum analysis, speech recognition, etc., which is the most frequently used digital signal processing algorithms. Base 2 FFT algorithm is the most common FFT algorithm, however, compared with it, base 4 FFT algorithm reduces the computational complexity and the number of iterations, which further improves the speed of the DFT. In theory, a larger base of FFT algorithm can further decrease the times of operation at the expense of the program or the complexity of the hardware. And the base 4 FFT algorithm which better balances the complexity and performance, is an ideal algorithm.

Some computer architecture with fused multiply-add instruction and these instructions have very important significance. First of all, on the software, it is execution time as single multiplication or adds instruction almost as fast; on the hardware, the FMA unit usually costs less than a single multiplier and adder. Then, there is no rounding operation between multiplication and addition, thus FMA can improve the calculation accuracy. Second, FMA instruction has the very high performance. For decoding rate on an average of one clock cycle one machine instruction, peak throughput of FMA instructions has two FLOPs, while the peak throughput of a single multiplication instructions or addition is one FLOPs. For different FFT program, although you can get the same result, the computing performance is not the same. Different architectures will have different mapping methods, so a lot of articles studied the FFT algorithm efficiently mapped to special computer architecture. In this paper, we describe the FFT algorithm based on the Kronecker product formula so that the FFT algorithm is well adapted to the architecture of YHFT-Matrix. Based on the architecture of vector processors YHFT-Matrix, use FMA algorithm for time domain sampling base 4 FFT algorithm to optimize and make full use of FMA instruction, so as to achieve the purpose of saving times and reducing the number of floating point operations, thus it improves the operation efficiency.

2 THE ARCHITECTURE OF YHFT-MATRIX

The kernel architecture of YHFT-Matrix is shown in Figure 1, and the structure includes a unified instruction fetch, distributed components, scalar processing unit (SPU), vector processing unit (VPU), the vector’s visit to the storage body, DMA, etc. Among them, the vector processing unit is carried out by the vector of 16 homogeneous unit (VPE), and vector execution
unit, through the code tree and mixed network data interaction, at the same time support 12 launch very long instruction word (VLIW) to develop instruction level parallelism and data level parallelism.

Then, sequence of N/4. Assuming that time sequence x(n) is decomposed into four long according to a serial number n, and the parity of the

Time-domain extraction (DIT) 4 FFT algorithm is

3mentation.

support for the vectorization of FFT algorithm implementation more convenient and efficient and provide good access, read and write access to 1024 bit vector data.

Then, the washing operation between VPE can be

are designed, with the wash and fetch needs for fusion. In the FFT algorithm, data rearrangement is needed in the local register of VPE, in order to reduce the washing mode and the influence of certificates of deposit to visit unit, and the special LOAD/STORE instructions are designed, with the wash and fetch needs for fusion. Then, the washing operation between VPE can be done through the LOAD/STORE operation, so as to make the FFT algorithm in the data interaction operation more convenient and efficient and provide good support for the vectorization of FFT algorithm implementation.

3 FFT ALGORITHM ANALYSIS

3.1 The principle of time-domain sampling base 4 FFT algorithm

Time-domain extraction (DIT) 4 FFT algorithm is according to a serial number n, and the parity of the time sequence x(n) is decomposed into four long sequence of N/4. Assuming that N=4^M, namely:

\[ x_0(r) = x(4r) \]
\[ x_1(r) = x(4r + 1) \]
\[ x_2(r) = x(4r + 2) \]
\[ x_3(r) = x(4r + 3) \]

Then:

\[ X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn} \]
\[ = \sum_{r=0}^{N/4-1} x(4r)W_N^{4kr} + \sum_{r=0}^{N/4-1} x(4r + 1)W_N^{4kr+1} \]

Such as the first decomposition, N point DFT is decomposed into four points of DFT, followed by the second, third..., decomposed step by step, until finally calculate the four point sequence of DFT.

As shown in Figure 2, a time-domain sampling based 4 FFT butterfly unit needs three complex multiplications and 8 times the plural addition. Considering one plural equals 4 times real multiplication and two real addition, once plural equals 4 times real multiplication and two real addition. So, a time-domain sampling base 4 FFT butterfly unit needs 34 times real floating point operations (12 times of multiplication and addition 22). With the number of floating point operations, measure the complexity of the algorithm, and the complexity of the N point DIT base 4 FFT is: \(4.25\ N\log2N\).

3.2 The definition of FMA

In order to effective utilize the advantage of the linear transformation, namely, the linear transformation algorithm is efficiently mapped to YHFT-Matrix architecture, such as FFT algorithm, which requires
the standard algorithm turns into FMA structure algorithm, so as to achieve the aim of making full use of FMA instructions. If given three input operands, the FMA instruction can be finished through any one of the following operations:

\[
\begin{align*}
  y &= a + b \times c \\
  y &= a - b \times c \\
  y &= a + b \times c \\
  y &= a - b \times c
\end{align*}
\]

Considering the linear transformation algorithm, only addition is the same (in this paper, considering the addition and subtraction) as multiplication (multiplied by a constant), so making the most of FMA instruction can make the minimum algorithm cost.

The purpose of using the FMA instruction optimization makes the transformation into: \(A = \text{diag}(i^4)A'\), \(A'\) is the perfect form of FMA. In the following, the related definition of FMA instruction is introduced.

Definition 1. Tensor product: It is the matrix block structured of \(A_{m \times n}, B_{k \times 1}\).

\[
A \otimes B = \begin{bmatrix}
  a_{0,0}B & \ldots & a_{0,n-1}B \\
  \vdots & \ddots & \vdots \\
  a_{m-1,0}B & \ldots & a_{m-1,n-1}B
\end{bmatrix}
\]

3.3 Using the FMA instruction to optimize DIT based 4 FFT

A time-domain extracting base 4 butterfly unit can be expressed as:

\[
\begin{bmatrix}
  A_{\text{out}} \\
  B_{\text{out}} \\
  C_{\text{out}} \\
  D_{\text{out}}
\end{bmatrix} = \mathcal{F}_4 \text{diag}(1, w_1, w_2, w_3)
\begin{bmatrix}
  A_{\text{in}} \\
  B_{\text{in}} \\
  C_{\text{in}} \\
  D_{\text{in}}
\end{bmatrix}
\]

And \(\mathcal{F}_4 = \begin{bmatrix}
  1 & 1 & 1 & 1 \\
  1 & -i & 1 & -i \\
  1 & 1 & -1 & -1 \\
  1 & -i & -1 & i
\end{bmatrix}\),

The decomposition \(\mathcal{F}_4 = (I_4 \otimes B_{2,4})(I_2 \otimes B_{2,2})R_4\)

\[
B_{2,4} = \begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 1 & i \\
  0 & 0 & i
\end{bmatrix},
\begin{bmatrix}
  1 & 1 \\
  1 & -1
\end{bmatrix},
\]

\[
R_4 = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

The time-domain extraction base 4 FFT butterfly unit can be expressed as

\[
\begin{bmatrix}
  A_{\text{out}} \\
  B_{\text{out}} \\
  C_{\text{out}} \\
  D_{\text{out}}
\end{bmatrix} = T_4
\begin{bmatrix}
  A_{\text{in}} \\
  B_{\text{in}} \\
  C_{\text{in}} \\
  D_{\text{in}}
\end{bmatrix}
\]

And \(T_4 = \begin{bmatrix}
  1 & 0 & w_1 & 0 \\
  0 & 1 & 0 & -iw_1 \\
  1 & 0 & -iw_1 & 0 \\
  0 & 1 & iw_1 & 1
\end{bmatrix}\)

After optimization of FMA instruction, calculating a time-domain extraction of the butterfly unit 4 FFT algorithm can be implemented by 24 FMA instructions. The computation complexity of N points FFT is \(3N\log_2N\). But, at the time of the specific implementation, rotating factor must be calculated in advance, according to the real part \((W_r)\) and imaginary /real part \((W_i/W_r)\) way of loading to the storage space.

3.4 Vectorization of FFT

In order to maximize the resource utilization of vector unit, exploit the data parallelism of FFT algorithm. Therefore, we need to solve the problem of good data storage, including the input data and storage of rotation factors.

3.4.1 The storage of input data

For the VM storage structure of YHFT-Matrix, the input data are placed in a row according to the real part and imaginary part of cross. Take 1024 double-precision floating-point as an example, which is shown in Table 1, every cell in the table is a 64 bits. 1024 input data, therefore, it needs to take up the storage size of \(2 \times 1024 \times 8\) byte = 16 KB.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
<td></td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>R[512]</td>
<td>I[512]</td>
<td>…</td>
<td>R[527]</td>
<td>I[527]</td>
</tr>
<tr>
<td>R[1009]</td>
<td>I[1009]</td>
<td>…</td>
<td>R[1023]</td>
<td>I[1023]</td>
</tr>
</tbody>
</table>

Table 1. 1024 DIT base 4 FFT data storage.

3.4.2 Storage of butterfly factor

According to the principle of extraction in time domain based 4 FFT algorithm, \(N=4^l\) point FFT algorithm requires M class to calculate. And in traditional computing, base 4 each of FFT butterfly unit needs to multiply by the rotation of the three different factors: \(W_N^i, W_N^{2i}\) and \(W_N^{3i}\). But, with the method of this article, 4 basic computational FFT butterfly unit just
calculate 2 different rotation factors: $W_N^i$ and $W_N^{2i}$, calculate the value of rotation factor, imaginary part and real part ($W_r^i$ and $W_r^{2i}$), and then put the rotation factors according to the real part ($W_r^i$), imaginary part and real part ($W_r^{2i}$) in cross storage way, which reduced the number of rotating factor storage. Vector access instructions can only access in a row, so every level FFT computation of different rotation factor should be placed separately.

Take 1024 points of DIT FFT base 4 as an example, the algorithm is realized by 5 butterfly calculation, and each level number required by different rotation factor are: 1, 4, 16, 64 and 256. Accordingly, the different rotation factor of level 1 and level 2 is less than the number of the count of VPE, and level 1 needs to multiply by the rotation of the factor value is 1, so it can be omitted. In order to improve the parallelism of computation, it can put the level 2 which the redundancy of the rotation factors needed for 4 times, that is, the storage of rotation factor, as shown in Table 2, and take up the size of the storage space for 11264 B.

<table>
<thead>
<tr>
<th>Series</th>
<th>Coefficient</th>
<th>Number of coefficients</th>
<th>Operation times</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>i=0,64,128,192</td>
<td>4*4</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>i=0,16,.....,240</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>i=0,4,.....,252</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>i=0,1,.....,255</td>
<td>256</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. 1024 DIT FFT base 4 rotation factor $W_{1024}^i$.

3.4.3 The selection of mixed washing way
For 1024 point time-domain extracting base 4 FFT operation, a total of 5 level operations are needed, but only level 1 and level 2 need data interaction between PE. As much as possible to reduce the demand and fusion to fetch request. Namely, it is through the special LOAD/STORE instructions to implement data interaction between VPE. First of all, we need two load operations for two pairs of words in the whole die 32 stored in situ. After calculation, the results are in situ. However, in the following, 4 FFT algorithms need to deposit in advance good rotation mode, and 2 factors are extracted. In the extraction of the input data, level 2, level 3 and level 1 do the same. For the convenience of the operation, after in calculating level 3 FFT arithmetic, it needs the results according to the real and imaginary part in situ storage way, namely the two STORE operation for two pairs of words in the whole die 32 stored in situ.

When making the level 4 and level 5 operation, we need two load operations for two pairs of words in module 2 extraction respectively, breaking the real and imaginary apart to convenient the calculate.

<table>
<thead>
<tr>
<th>Series</th>
<th>Coefficient</th>
<th>Number of coefficients</th>
<th>Operation times</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>i=0,64,128,192</td>
<td>4*4</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>i=0,16,.....,240</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>i=0,4,.....,252</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>i=0,1,.....,255</td>
<td>256</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3. 1024 DIT FFT base 4 rotation factor $W_{1024}^{2i}$.

Figure 3. The overall mode 4 extract.

Figure 4. The overall mode 32 extract.

Figure 5. Mode 2 respectively extracted.
By washing and fetch fusion, can effectively reduce the washing mode’s influence on the fetch, and also to prevent excessive mixed wash times and extra storage space.

3.4.4 The assembly instruction optimization

In FFT assembler, butterfly operation can be done through three nested loops. The first loop controls cycle level, the second loop controls number of butterfly unit, and the third loop controls a single butterfly unit operation. In order to fully exploit instruction-level parallelism, we need software pipelining of the program. This involves the writing of the software running water table, the determination of iteration interval, data correlation, resource constraints and problems, such as the life cycle of data.

For 1024 points DIT base 4 FFT, it needs total 5 operations, and calculating a butterfly unit needs 12 times multiplication and 22 plus (minus) operation (except level 1), before using FMA instruction optimization, namely need 34 MAC. That is the FFT operation at level 1, due to the rotation of the need to take a factor of 1, calculating a butterfly unit only needs 16 MAC and eight LOAD/STORE. So make sure the minimum interval iteration is 6, and MAC utilization rate is 88.9%. In level 2-5 FFT arithmetic, calculating a butterfly unit needs 34 MAC and 11 LOAD/STORE, so make sure the minimum iteration interval is 12, when the MAC utilization rate is 94.4%. But it is important to note that because the first operand is not immediately involved in calculation, it will be more than its life cycle, which should be extended. After the FMA instruction was used to optimize, calculating a butterfly unit can be implemented by 24 FMA instructions and demands 24 MAC. In level 1 FFT arithmetic, calculating a butterfly unit needs only 16 MAC, which determines the minimum interval iteration to be 6. In level 2-5 FFT arithmetic, calculating a butterfly unit needs MAC 24 and 10 LOAD/STORE (including the number of four operations, two rotation factors, and the result of the four data). So, it meets the resource constraints, such as data correlation conditions under the premise of determining the minimum interval iteration of 8, and the utilization rate of MAC is 100%.

In this way, after optimization of the FMA instruction, improve the efficiency of MAC parts, and then improve the computation efficiency of the chip. At the same time, through the full software pipelining, FFT program improve the instruction level parallelism, so as to improve the computing performance of the program. And, for the time domain sampling, base 4 FFT arithmetic, also saves storage space of the rotation factor.

4 EXPERIMENTS

With the YHFT-Matrix vector processor test simulation platform, respectively tested 64, 256, 1024, 4096, 16384, 65536 double-precision floating-point plural time-domain extracting base 4 FFT algorithm, estimate the performance and computational efficiency with or without FMA optimization, and compare the performance with TIC6713. The result is shown in Figure 6 and Figure 7.

As can be seen from Figure 6, when the point is small, the speedup is not obvious before the optimization of FMA and after the optimization of FMA. This is mainly due to the FMA when software pipelining optimized FFT program filling time is longer than FFT without FMA optimization filling, the number of loop is less, which leads that the optimization effect is not obvious.

As can be seen from the Figure 7, before the FMA optimization, when the point number is 64, the computational efficiency is only 14.91%. When points are up to 4096 points, the calculating efficiency is 45.20%. With the increase of points, the calculation efficiency has promoted, but much of ascension. And if the point of FFT computation is very big, it will bring the cache invalidation problem. Therefore, big points FFT does not generally calculated directly, but to take the big points FFT into line with the Cache size calculated points FFT method. And after the FMA optimization, when the point number is to 64, the computational
efficiency which compared with no FMA optimization is improved by about 3%, and with the increase of points, the computational efficiency has been sharply improved. When the points rise to 4096 points, compared FMA before optimization, computational efficiency has been increased by 13.36%; when points rise up to 65536, computational efficiency is from 17.71% by 64, up to 62.31%.

5 CONCLUSION

This article in the light of the vector processors with FMA structure, making full use of FMA instruction to optimize the time-domain extracting 4 FFT algorithm, and to realize the FFT algorithm more efficiently. By the method of software pipelining to optimize the core code of the manual assembly, compared with no use of FMA instruction optimization of time-domain extracting 4 FFT algorithm, and the computational efficiency and performance of the chip is also improved.

ACKNOWLEDGEMENT

This paper is financially supported by the National Natural Science Foundation of China (61133007 & 61303065) and Doctoral Fund of Ministry of Education (20134307120028).

REFERENCES


