Hardware-Efficient Index Mapping for Mixed Radix-2/3/4/5 FFTs

Citation

Year
2016

Version
Peer reviewed version (post-print)

Link to publication
TUTCRIS Portal (http://www.tut.fi/tutcris)

Published in

DOI
10.1109/SAMOS.2016.7818348

Take down policy
If you believe that this document breaches copyright, please contact tutcris@tut.fi, and we will remove access to the work immediately and investigate your claim.
Hardware-Efficient Index Mapping for Mixed Radix-2/3/4/5 FFTs

Tomasz Patyk  
Dolby Poland  
Wroclaw, Poland  
Email: tomasz.patyk@dolby.com

Jarmo Takala  
Tampere University of Technology  
Tampere, Finland  
Email: jarmo.takala@tut.fi

Abstract—Orthogonal frequency-division multiplexing modulators and demodulators for modern communication standards require efficient implementation of the fast Fourier transform (FFT). Traditionally, radix-2 and radix-4 FFT algorithms have been used. Over the last few years, support for non-power-of-two transform sizes, with the emphasis on the radix-3 and radix-5, started to become a standard. We have created a systematic approach for designing simple digital circuits that compute array access indices for the mixed radix-2/3/4/5 FFT computations. Proposed index mapping, allows for the use of a bit rotation instead of the add/modulo and multiply operations. Index generation circuits, implementing the proposed index mapping, have hardware complexity comparable to general index generation circuits for power-of-two FFTs.

I. INTRODUCTION

In the year 1965, Cooley and Tukey presented their seminal paper [1] and brought the fast Fourier transform (FFT) algorithm to scientific and engineering communities’ attention. Their work, laid the groundwork for a fledgling discipline of digital signal processing (DSP). Since then, the FFT along with digital filters constitute two most important classes of DSP algorithms [2].

The attractiveness of the FFT algorithm comes from the computational complexity reduction it offers. The quadratic \( O(N^2) \) complexity of the discrete Fourier transform (DFT) calculated by definition, is reduced to a linearithmic \( O(N \log N) \) complexity. Cooley and Tukey [1] proved that for a given radix, complexity is proportional to \( N \log N \). Compared to a direct calculation of the DFT, FFT algorithms exploit two techniques to avoid redundant operations: the divide and conquer technique, and a short length DFT optimisation. By employing the divide and conquer approach and reorganising data, a large sized DFT is recursively divided into smaller ones. This was extensively described by many authors [1], [3], [4], [5] and generalised to multidimensional index mapping by Burrus [6].

Two classes of factorisation divide FFT algorithms into prime factor algorithms (PFA) and common factor algorithms (CFA). Optimisation of a short length DFT is most commonly known for power-of-two DFTs of size 2 and 4 (radix-2 and radix-4 butterflies) which can be implemented without multiplication and with only additions. Winograd [4] presented optimised DFTs of non-power-of-two sizes of 3, 5, and 7.

Historic applications of the FFT include spectral analysis, filter banks, convolutions, and many more [2]. However, recent years have put a spotlight on the FFT use in communication applications. In particular, orthogonal frequency-division multiplexing (OFDM) which is used in wideband digital communication networks including 3G and 4G mobile communication networks. Efficient OFDM modulator and demodulator implementations are often based on the IFFT and FFT computations [7]. One of the challenges that the FFT has to face in the new field of application is an efficient implementation of non-power-of-two DFT sizes. As the uplink precoding [7] of Long-Term Evolution (LTE) requires transform sizes of 12 – 1296, it is clear that efficient implementations of mixed radix algorithms, including radices 3 and 5, will gain popularity. Challenges are yet to overcome as non-power-of-two computations of odd radices are not trivial to implement on binary logic, which we use today.

Linear mapping of data indices from one-dimensional to multi-dimensional mapping reduces the numbers of multiplications and additions required to calculate a DFT. However, performing linear mapping itself can be non-trivial task taking up significant chunk of hardware resources or computational time [8]. Therefore, non-power-of-two DFT sizes and algorithms with regular access patterns have been preferred over the years. Efficient hardware implementations for single radix-2, radix-4, and mixed-radix-2/4 are reported in the literature. On the contrary, number of hardware implementations for non-power-of-two DFT sizes is limited. Typically those papers present extra hardware for add/subtract and modulo operations [9] or complex multiplications [10].

In this paper, we propose a novel hardware efficient array index mapping for a decimation-in-time (DIT) mixed-radix-2/3/4/5 FFT algorithms to be used for in-place computations. The scheme is generic as it supports any DFT size which can be factorised to supported radices. The order of radices can be chosen arbitrarily. Additionally, we present an implementation of an index generator circuit (IG). The implementation is based on pseudo-linear counter and rotators, and it does not require any multipliers or extra adders. Small design corresponds to less silicon area, shorter critical path, and reduced power consumption. Latter is especially important for battery-powered portable devices where new communication
The rest of the paper is organised as follows. Section II presents current state of the research in the field. Section III explains the novelty of the proposed index mapping. Section IV proposes a hardware efficient implementation of the new mapping. Section V discusses advantages of the design as compared to existing solutions. Finally, section VI concludes the paper.

II. RELATED WORK

There are several aspects of a FFT algorithm that translate to different index mapping. For a given DFT size, there are decimation-in-time [1] or decimation-in-frequency [3] algorithms; input sequence in order or permuted; for a mixed radix FFT, the order of radices can be chosen arbitrary, and finally, a computation stage can have regular or irregular (split-radix) geometry [5]. Demuth [11] proposed an unified set of equations, that allow FFT computations of an arbitrary DFT size. The implementation based on three nested loops can be implemented on programmable processor using general purpose arithmetic logic unit (ALU). Naturally, using ALU for index computations usually means that ALU does not compute FFT kernel (butterfly). This approach is flexible but far from optimal at the same time.

FFT implementations based on application-specific integrated circuits (ASIC) or instruction-set processors (ASIP) tend to use dedicated index (address) generation units (IG). Hardware-efficient IG units have been proposed for power-of-two FFTs with regular signal flow graphs, both single radix [12] and mixed radix [13]. These design transform a linear index generated with an accumulator to the actual array access index through a bit rotation, where the number of rotated bits depends on the computational stage in the FFT. The rotations can be implemented with a set of multiplexers making the design as hardware-optimal.

Efficiency of hardware implementation for power-of-two FFTs is related closely to binary logic; Indices are obtained with multiplications/divisions by power-of-two factors, which can be efficiently implemented with shifters. This is not the case for non-power-of-two radices. For instance single radix-3 FFT will require multiplication by power-of-three numbers. For mixed radix-3/4 FFT index calculations will require multiplication by integer multiples of 12.

Few authors [14], [15], [16], [17] proposed a LTE compliant designs, which support DFT sizes ranging power-of-two as 128–2048 and 1536. However, 1536 can be considered as a special case as it contains a single radix-3 stage, which can be left last in the computations, thus the ordinary power-of-two index mapping hardware can be used.

Hsiao [9] proposed a generalised, mixed radix algorithm and its hardware implementation. Address (index) generation hardware unit is a direct implementation of the index mapping proposed by Burrus [6]. All indices for a single butterfly are generated in parallel. This requires two accumulators, adder, and a modulo circuit per butterfly input. For each stage
different seed values need to be provided for the accumulators. In addition, modulo logic must be set to DFT size. Each index is obtained by adding together outputs of two accumulators and modulo truncated if needed. The authors did not explain, if the seed values required for accumulators, are calculated on the fly or stored in a ROM memory and fetched by the control logic. Chen [18] extends the previous method to support wider range of radices. The extended scheme also supported several DFT sizes using the same address (index) generation hardware.

Ma [10] proposed an approach, where pseudo-linear address from accumulator is first rotated. The number of rotated bits depends on the stage and decreases as computation proceeds. Certain chunks of the accumulator are then multiplied by constant values. The products of those multiplications are added together to obtain the final index. For longer DFT sizes, this approach might become extremely expensive as it requires \((s - 1)\) multiplications and additions for \(s\) computational stages.

The proposed index mapping supports non-power-of-two radices and mixed radix FFTs. Unlike the previous solutions discussed earlier it requires neither add/modulo or multiplication to obtain memory access indices.

### III. PROPOSED INDEX MAPPING

The DFT of an input vector \(Y = [x_0, x_1, ..., x_{N-1}]^T\) is defined as the vector \(Y' = [y_0, y_1, ..., y_{N-1}]^T\) such that:

\[
y_m = \sum_{n=0}^{N-1} \omega_{N}^{nm} x_n,
\]

or equivalently \(Y = F_{N} X\), where \(F_{N}\) is the \((N \times N)\)-matrix of DFT with entries:

\[
\omega_{N}^{nm} = \exp \left( -\frac{i 2 \pi nm}{N} \right).
\]

Many FFT algorithms were derived for efficient computation of the DFT. In this paper, we use the in-place, decimation-in-time (DIT), mixed radix algorithm with permuted input, and in-order output. The formula for this FFT algorithm is given by (3):

\[
F_{p_t} = \prod_{s=m}^{1} \left\{ I_{p_{t+1}} \otimes (F_{r_s} \otimes I_{p_{t-1}}) \left( I_{p_{t+1}} \otimes T_{r_s,p_{t-1}} \right) \right\} R_{p_t}.
\]

where \(I_N\) denotes an identity matrix of order \(N\), \(\otimes\) is a tensor product, \(m\) is number of stages, \(s\) is the stage index in \(s \in (1, m)\), \(r_s\) is the radix of the \(s^{th}\) stage, and \(p_{m}^{n}\) is a product of radices of stages \(n\) to \(m\) given by:

\[
p_{m}^{n} = \begin{cases} 
I_{n} & \text{if } n \leq m, \\
1 & \text{if } n > m.
\end{cases}
\]

\(T\) matrix of stage twiddle factors is given by:

\[
T_{r,k} = \oplus_{i=0}^{r-1} D_{N}^{k,i},
\]

\[
D_{N}^{k,i} = \text{diag}(\omega_{N}^{0i}, \omega_{N}^{1i}, ..., \omega_{N}^{(k-1)i}),
\]

where \(N = r \times \) equals DFT size and \(\oplus\) denotes the matrix direct sum. Finally, \(R_{p_t}^{n}\) is an input permutation matrix:

\[
R_{p_t}^{n} = \prod_{s=m}^{1} \left( (I_{p_{t+1}}^{m} \otimes P_{p_{t-1},r_s}) \right)
\]

based on the stride-by-\(K\) permutation matrix [19] of order of \(N\) defined as follows:

\[
[P_{N,K}]_{m,k} = \begin{cases} 
1 & \text{if } k = (mK \mod N) + [mK/N], \\
0 & \text{otherwise.}
\end{cases}
\]

### TABLE II

<table>
<thead>
<tr>
<th>Linear</th>
<th>All stages</th>
<th>Stage 1</th>
<th>Access Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-10</td>
<td>R-3</td>
<td>R-3</td>
<td>R-3</td>
<td>R-10</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>002</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>010</td>
<td>010</td>
<td>3</td>
<td>001</td>
</tr>
<tr>
<td>4</td>
<td>011</td>
<td>011</td>
<td>4</td>
<td>101</td>
</tr>
<tr>
<td>5</td>
<td>012</td>
<td>012</td>
<td>5</td>
<td>201</td>
</tr>
</tbody>
</table>

### TABLE III

<table>
<thead>
<tr>
<th>Linear</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-10</td>
<td>R-3/4/3</td>
</tr>
<tr>
<td>R-10</td>
<td>R-3/4/3</td>
</tr>
<tr>
<td>Stage 2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>00000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>00002</td>
</tr>
<tr>
<td>3</td>
<td>00003</td>
</tr>
<tr>
<td>4</td>
<td>0010</td>
</tr>
<tr>
<td>5</td>
<td>0011</td>
</tr>
<tr>
<td>6</td>
<td>0012</td>
</tr>
<tr>
<td>7</td>
<td>0013</td>
</tr>
</tbody>
</table>

The signal flow graph of a 32-point mixed radix-4/4/2 FFT is illustrated in Figure 1. The algorithm uses in-place processing; input data is read through a stride permutation, processed in stages, and overwritten by the result data. Thus,
the same index mapping can be used for reading and writing. Data fed to the first stage is shuffled with (7) while the output data is in order. The array access indices for $s^{th}$ radix stage, defined in (3) by tensor products, can be represented with a permutation matrix as:

$$I_{p_{s+1}} \otimes P_{r_{p_{s-1}}p_{s-1}}.$$  \hfill (9)

The equation (9) implies that index computations require multiplication, division, modulo operation, and addition. However, for stride-by-$K$ matrix of order $N$, where both $K$ and $N$ are power-of-two values, the access address can be obtained with a rotation of linear binary index [12], [13]. For stage $s$, $\sum_{i=1}^{s} log2r_i$-least significant bits are rotated right by $log2r_s$ bits. Linear indices rotated to get access indices of the 32-point FFT are presented in Table I. Bits $b_1b_0$ and $b_3b_2$, correspond to radix-4 stage 1 and 2, respectively. Bit $b_1$ represents stage 3 of radix-2. The bits of the linear index indicated in bold are the indices by rearranging all radices to the order in which their corresponding to order of radices in the FFT computations. Rotating the index alone does not produce expected access indices anymore.

In this paper, we propose to use linear index in binary-coded mixed radix (BCMR) representation. BCMR encoding, unlike binary-coded decimal (BCD), uses $n = \lfloor log2r_s \rfloor$-bits to represent a numerical digit. Contrary to BCD, radix, and therefore number of bits representing digits in the BCMR, might be different for each numerical position. For the linear index, radices assigned to numerical position change with the stage. The radix of the least significant digit $r_s$ is the radix of current stage $s$. Other digits have radices assigned according to the order of radices in the FFT computations. Rotating linear indices produces an unambiguous mapping to access indices by rearranging all radices to the order in which they are computed. Tables IV and V presents BCMR-based index mappings for the 27- and 144-point FFTs, respectively.
Comparison of the BCMR-based indices with corresponding decimal indices in Tables II and III reveals that the proposed mapping, nonetheless unambiguous, is discontinuous in the address space. As a result, a larger memory space is needed than the DFT size indicates. In practice, however, FFT processing units support many DFT sizes; if the largest supported power-of-two DFT uses \( n \) bits for access indices, all non-power-of-two DFTs that use \( n \) bits for addressing, are supported too.

Figure 2 gives an example of a signal flow graph for 18-point FFT (multiplications by twiddle factors shown on the figure). This input could be omitted if the largest supported DFT size is smaller than the output size.

Figure 3. Signal flow graph of 30-point FFT (multiplications by twiddle factors omitted for clarity).

IV. HARDWARE IMPLEMENTATION

The index mapping proposed in section III can be efficiently implement in hardware. The unit is comprised of two blocks: the mixed radix accumulator and the rotator. The rotator converts the linear index produced by the accumulator to the array access index.

Figure 4 depicts an example circuit designed for the 30-point FFT presented in Figure 3. The unit has two inputs and a 6-bit output to read access indices from. The 1-bit \( \text{trig} \) input, if set to 1, triggers unit to produce consecutive access indices on each clock cycle. The \textit{stage} input sets internal multiplexers and demultiplexers through a simple logic (not shown on the figure). This input could be omitted if the unit had internal stage counter. The \textit{rotator} is build from six, 6-input multiplexers. The \textit{mixed radix accumulator} has three radix accumulator blocks, one per stage. Each block is connected to the \textit{trig} input and other blocks through a set of multiplexers and demultiplexers. Those are configured for \( s^{th} \) stage such that \( r_s \) accumulator block is set to be the first, and remaining blocks are connected in the order of computations. The \( \text{radix} – 1 \) values must be set to blocks on initialisation and remain unchanged during the FFT computations.

An example of a \( \text{radix} – 3/4 \) accumulator block (R3/4), a building block of the \textit{mixed radix accumulator}, is given in Figure 5. When input \( c_{in} \) is set to 1, the \textit{incrementer} increases its value by 1 on each clock cycle. When \textit{comparator} detects \( \text{radix} – 1 \) value in the registers, \( 0s \) are written to them in the next clock cycle.

This design can be easily extended to any DFT size and radix configuration if corresponding \textit{radix accumulator blocks} are present. Multiple DFT sizes can be supported when simple logic controlling multiplexers and demultiplexers is used. A design with \( s \) radix accumulator blocks requires: \( s \)-input
multiplexer and 2-output demultiplexer for the first block; 3-input multiplexer for the last block; 2 or 3-input multiplexer and 2-output demultiplexer for the next to last block; and 3-input multiplexer and 3-output demultiplexer for all the remaining blocks.

V. COMPARISON

As explained in section IV, the implementation of the proposed index mapping for n-bit addressing space requires n flip-flops, a set of multiplexers connecting them, simple comparators, and n multiplexers for bit rotation. Compared to implementations [12] and [13], which only support power-of-two DFT sizes, connecting multiplexers and comparators were added. When the number of supported radices was extended beyond power-of-two. Designs presented in [9] and [18] support odd radices but at the cost of two accumulators, adder, and modulo operation implemented with a subtractor and a multiplexer. They can obtain several addresses in parallel by duplicating hardware resources while this can be done also with the proposed method. Finally, the design in [10], alike the proposed method, uses bit rotation, but partial results require costly multiplications and additional addition.

VI. CONCLUSION

In this paper, we proposed a novel array index mapping for mixed radix-2/3/4/5 FFTs. The mapping allows for the use of a bit rotation, instead of add, modulo and multiply operations, to obtain indices for memory accesses in the FFT computations. A systematic method for designing a hardware-efficient implementation was discussed. Address generation routine is simple and requires small silicon footprint. The proposed method can be used as a memory addressing unit, in an application-specific fixed-function FFT processor or a address generator unit of a application-specific instruction-set processor (ASIP), for low-power and fast FFT computations.