Multiplierless Unity-Gain SDF FFTs

Mario Garrido Gálvez, Rikard Andersson, Fahad Qureshi and Oscar Gustafsson

Journal Article

N.B.: When citing this work, cite the original article.

©2016 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

http://dx.doi.org/10.1109/TVLSI.2016.2542583
Postprint available at: Linköping University Electronic Press

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-132483
Multiplierless Unity-Gain SDF FFTs

Mario Garrido, Member, IEEE, Rikard Andersson, Fahad Qureshi and Oscar Gustafsson, Senior Member, IEEE

Abstract—In this paper we propose a novel approach to implement multiplierless unity-gain SDF FFTs. Previous methods achieve unity-gain FFTs by using either complex multipliers or non-unity-gain rotators with additional scaling compensation. Conversely, this paper proposes unity-gain FFTs without compensation circuits, even when using non-unity-gain rotators. This is achieved by a joint design of rotators so that the entire FFT is scaled by a power of two, which is then shifted to unity. This reduces the amount of hardware resources of the FFT architecture, while having high accuracy in the calculations. The proposed approach can be applied to any FFT size and various designs for different FFT sizes are presented.

Index Terms—FFT, Unity-Gain, Multiplierless, Pipelined Architecture, SDF, CORDIC, CCSI

I. INTRODUCTION

In today’s digital signal processing world, there is often a need to convert signals between time and frequency domain. For this reason, the fast Fourier transform (FFT) has become one of the most important algorithms in the field. In order to calculate the FFT efficiently, various hardware architectures have been proposed. When high performance is required, feedback [1]–[8] and feedforward [9], [10] hardware FFT architectures are attractive options, as they offer high throughput capabilities.

Single delay feedback (SDF) FFT architectures [1]–[8] consist of a series of stages that process one sample per clock cycle. Each stage contains a butterfly and a rotator. The butterfly calculates additions and the rotator carries out rotations in the complex plane by given rotation angles, called twiddle factors [11].

Compared to the additions of the butterfly, rotations are more costly operations. For this reason, different approaches to implement rotators have been proposed in the past. The most straightforward approach is to use a complex multiplier [12], which consists of four real multipliers and two adders. In addition, it requires a memory to store the twiddle factors. Another option is to implement the rotators as shift-and-add operations. Following this idea, the CORDIC algorithm [13]–[16] breaks down the rotation angle into several successively smaller angles and rotates each of them with a fixed shift-and-add network. Another alternative is to use multiplier-based shift-and-add rotators [11], [17]–[20]. By using techniques such as Multiple Constant Multiplication (MCM) [21]–[23], these rotators carry out the rotation by reducing the complex multiplier to shift-and-add operations.

Among all these alternatives, multiplier-based shift-and-add rotators [11], [17]–[20] are the most efficient option for small set of twiddle factors, whereas CORDIC-based rotators [13]–[16] are the best alternative for large ones. However, CORDIC-based approaches and some multiplier-based shift-and-add rotators [11], [19] scale the output by a scaling factor $R \neq 1$. This scaling allows for more accurate and hardware-efficient rotations [12]. In order to achieve unity-gain, previous works have compensated the scaling factor by adding a scaling stage to the rotator [14]–[16]. However, this increases the usage of hardware resources.

In this paper we present novel multiplierless unity-gain SDF FFTs. They are obtained by designing the rotators in all FFT stages simultaneously, so that the output of the FFT has unity gain. Thus, the proposed approach neither requires the use of costly unity-gain rotators, nor circuits to compensate the scaling. This reduces the complexity of the FFT rotators and guarantees unity-gain for the FFT. In this paper we study different FFT sizes and propose suitable solutions for each size. Comparison in terms of number of adders and experimental results show the advantages of the proposed approach with respect to previous works.

The paper is organized as follows. In Section II we give an introduction to the SDF architecture and the FFT twiddle factors. In Section III we explain the proposed approach and provide optimized architectures for different FFT sizes. In Section IV we compare the proposed architectures to previous ones. In Section V we present the experimental results and in Section VI we summarize the main conclusions of the paper.

II. BACKGROUND

A. The SDF FFT Hardware Architecture

The SDF FFT is one of the most attractive and widely used FFT architectures. It allows for high throughput and requires a low amount of resources. An example of SDF FFT for $N = 64$ points is shown in Fig. 1. It consists of $n = \log_2(N)$ stages. Each stage includes a radix-2 (R2) butterfly and a rotator. The internal structure of a SDF stage is shown in Fig. 2. It consists of a butterfly, two complex multiplexers, a buffer, a rotator and a rotation memory. Additional pipelining registers can be used to reduce the critical path and increase the throughput.

B. The FFT Twiddle Factors

In the FFT rotators, each input is rotated by a different angle. This angle is determined by the twiddle factor...
obtains multiplierless unity-gain SDF FFTs. The method is

B. Design Method

TABLE I
TWIDDLE FACTORS OF A 64-POINT DIF FFT FOR DIFFERENT RADICES.

<table>
<thead>
<tr>
<th>Radix</th>
<th>FFT Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Radix-2</td>
<td>$W_{64}$</td>
</tr>
<tr>
<td>Radix-2$^2$</td>
<td>$W_4$</td>
</tr>
<tr>
<td>Radix-2$^3$</td>
<td>$W_4$</td>
</tr>
<tr>
<td>Radix-2$^4$</td>
<td>$W_4$</td>
</tr>
</tbody>
</table>

$W_L^\phi = e^{-j2\pi\phi/L}$. The parameter $L$ is constant for each stage and represents the number of different rotation angles in that stage. These angles are the result of dividing the circumference in $L$ equal parts. Among them, the specific rotation angle is determined by the parameter $\phi$, which is a natural number in the range $[0, L-1]$.

The complexity of the rotator depends on the number of angles that it has to rotate, $L$. The simplest rotator is $W_L = W_4$. This twiddle factor only includes trivial rotations ($0^\circ$, $90^\circ$, $180^\circ$ and $270^\circ$). Trivial rotations are characterized by the fact that they can be calculated by simply exchanging the real and imaginary part of the inputs and/or changing their sign.

For power-of-two FFT sizes, $L$ is also a power of two and its value ranges from 4 to $N$. The specific value for each stage depends on the radix and decomposition of the FFT. The most typical decompositions are decimation in time (DIT) and decimation in frequency (DIF) [24]. The most common radices nowadays are radix-2$^k$ [9], [25]. Table I shows typical layouts for the 64-point DIF FFT in Fig. 1. Note that the twiddle factors for radix-2$^2$, 2$^3$ and 2$^4$ are simpler than those in radix-2, which leads to simpler rotators. Note also that radix-2$^k$ SDF architectures only differ in the twiddle factors. The rest of the SDF architecture is independent of the algorithm.

III. PROPOSED APPROACH

A. Problem Formulation

The research problem solved in this paper can be formulated as follows. On the one hand, when implementing an FFT we can choose among a large number of FFT algorithms [25]. On the other hand, there are numerous approaches to implement the rotators that calculate each twiddle factor. These implementations need different number of adders and have different gains. However, we aim for unity gain, so that the FFT outputs are not scaled. The research question is then: How to design an SDF FFT with the smallest number of adders and unity gain?

B. Design Method

Based on the previous considerations, the following method obtains multiplierless unity-gain SDF FFTs. The method is based on a joint selection of the FFT algorithm and the twiddle factors of the FFT stages.

Step 1: Select suitable FFT algorithms. All the possible FFT algorithms are discussed in [25]. These algorithms differ in the twiddle factors at the different FFT stages, as shown in Table I. The selection of a suitable algorithm is done based on the twiddle factors and on the type of rotators that are involved. Regarding the type of rotators, CORDIC-based approaches are nowadays the best for large twiddle factors ($W_{64}$ and larger), and CCSSI for small twiddle factors ($W_8, W_{16}$ and $W_{32}$) [11]. Both of them lead to multiplierless rotators, i.e., they only use shift-and-add operations.

The criterion used to select the algorithm is to minimize the number of large twiddle factors calculated by the CORDIC, which are the most costly ones, and maximize the number of trivial rotators ($W_4$), which are the cheapest ones. There are also trade-offs between the number of big ($W_{64}$ and larger) and small ($W_8, W_{16}$ and $W_{32}$) twiddle factors. In such case, various FFT algorithms are considered. For instance, a 1024-point radix-2$^5$ FFT includes 1 large twiddle factor and 4 small ones, and a 1024-point radix-2$^4$ FFT has 2 large twiddle factor and 2 small ones. Here, both cases are attractive.

Step 2: Determine the scaling of the rotators. This is done by considering that the total scaling of the FFT is a power of two, which can be reduced to unity by just considering that the binary point is in a different position of the binary representation [11], i.e,

$$R_{tot} = \prod_s R_s \approx 2^q, \quad q \in \mathbb{Z}. \quad (1)$$

The scaling of the CORDIC is constant and cannot be modified. Thus, the scaling of those stages in which the CORDIC is used is considered to be $R_s = R_C$, where $R_C$ is a constant. Specifically, $R_C = 1.647$ in the conventional CORDIC [13] and $R_C = 1.164$ in the memoryless CORDIC [16], when considering seven or more micro-rotation stages. For trivial rotators ($W_4$), the scaling is $R_s = 1$. As a result, in order to achieve unity gain for the FFT, the scaling for small rotators needs to be

$$\prod_{s^*} R_s = \frac{2^q}{(R_C)^{N_C}}, \quad q \in \mathbb{Z}, \quad (2)$$

where $N_C$ is the number of stages where CORDIC rotators are used, and the product over $s^*$ refers to the stages in which small rotators are used.

Step 3: Design the rotators for the small twiddle factors, so that the total scaling fulfills (2). This is done by applying the method for the CCSSI rotators [11] in the following way. First, each of the rotators is designed considering the case of a multiple constant rotator (MCR) with uniform scaling. This means that the scaling of each of the rotators is not fixed yet and a list of candidates for each rotator is obtained. Then, different combinations of rotators for the different stages are evaluated in terms of number of adders, rotation error and $R_{tot}$. This provides the most efficient designs.

Step 4: Select the architecture that offers the most suitable trade-off between area and accuracy. Apart from the rotation error, in this step other accuracy measures such as the SQNR.
or the Frobenius norm [12] may be evaluated depending on
the demands of the target application.

C. Proposed Unity-Scaled FFT Architectures

Table II shows the proposed unity-gain SDF FFT architec-
tures. The algorithm is defined by the twiddle factors at each stage. \( T_4 \)
stands for a trivial rotation, while \( M_L \) and \( C_L \) refer to CCSSI
and CORDIC rotators of size \( L \), respectively. For example, the
algorithm \( T_4 M_{16} T_4 C_{64} T_4 \) has a \( W_{16} \) CCSSI rotator
at stage two and a \( W_{64} \) CORDIC at stage four. The \( M_L \) rotators
used in the architectures for \( W_8, W_{16} \) and \( W_{32} \) are
detailed in columns two to six. They show the number of adders of the entire FFT
architecture. The total number of real adders for the FFT is calculated as

\[
\text{FFT Adders} = 4 \log_2 N + 2S_C N_C + \sum \text{Add}(W_8, W_{16}, W_{32}) \quad (3)
\]

The first term is the number of adders in the butterflies, i.e., 4 real adders per radix-2 butterfly. The second term is the number of adders for the CORDIC rotators. In the architecture we use \( S_C = 10 \) stages of micro-rotations for the memoryless CORDIC [16], with two adders per stage. Thus, the last micro-rotation coefficient is \( 1 \pm 2^{-9} \), which represents an angle of \( \alpha = 0.11^\circ \). This resolution is enough for up to 1024-point FFTs, where the minimum rotation angle is \( 360^\circ / 1024 = 0.35^\circ \). For larger FFTs, more CORDIC stages can be considered, with the additional adder cost. The last term corresponds to the adders for the \( W_8, W_{16} \) and \( W_{32} \) rotators shown in the table.

The last column of Table II shows the FFT gain, \( G_{FFT} \). This is equal to \( R_{tot} \) normalized to unity by dividing it by the closest power of two:

\[
G_{FFT} = \frac{R_{tot}}{2^{\text{round}(\log_2 R_{tot})}} \quad (4)
\]

In all the proposed designs, the FFT gain is approximately equal to 1, which meets the requirement of unity scaling. Furthermore, the proposed FFTs do not use complex multipliers, as all the rotators are implemented as shift-and-add.

The table offers a trade-off between total number of adders and rotation error. For instance, the proposed radix-\( 2^4 \) 512-point FFT has low rotation error in all its rotators and requires a total of 94 adders. By contrast, the radix-\( 2^4 \) 512-point FFT has higher \( \epsilon \) in some rotators, but only requires 80 adders. Thus, the proposed method offers a trade-off between area and accuracy.

IV. COMPARISON

Table III compares the proposed designs to previous SDF
FFT architectures, where all of them process one sample
per clock cycle in pipeline. The table includes the hardware
resources for a general \( N \), as well as for the particular case
of \( N = 1024 \). Some previous designs use advanced radices
such as radix-\( 2^2 \) and radix-\( 2^4 \), and use complex multipliers for
the rotations [1]–[4]. This requires four complex multipliers
per rotator [1], [2]. Alternatively, three real multipliers per
rotator have been used at the cost of increasing the number of
adders [3], [4]. Other previous designs propose multiplierless
solutions that remove the overhead of the multipliers by only
using shift-and-add operations for the rotators [5], [6].

The comparison shows that the proposed multiplierless
unity-gain FFTs need the smallest number of adders among
previous SDF FFT architectures. This holds even when substituting
multiplier-based rotators by CORDIC rotators in current
efficient designs (*).

V. IMPLEMENTATION

The two designs marked with a star (**) in Table II have been
described in VHDL. The architecture chosen for 256 points
has the lowest rotation error and the advantage of reusing the
same \( W_{16} \) rotator in two stages. For 1024 points, the radix-\( 2^4 \)
1024-point FFT has been chosen because it reuses the same
\( W_{16} \) rotator and the rotation error is smaller than that of the
\( W_{32} \) rotator in the radix-\( 2^5 \) FFT.

![Diagram](image-url)

Fig. 3. Internal structure of the CCSSI \( W_{16} \) rotators in the implemented SDF
FFTs. (a) Calculation of the products for the rotator \( \{168, 155 + j/64, 119 + j/119\} \)
in the 256-point SDF FFT. (b) Calculation of the products for the
rotator \( \{311, 288 + j/119, 220 + j/220\} \) in the 1024-point SDF FFT. (c) Combination
of the products for both rotators.

Figure 3 shows the internal structure of the rotators used in
both designs. Both of them consist of six adders, and 14 and
16 multiplexers, respectively. The shifts shown in the figure
are hard wired and, thus, they do not have any hardware cost.
Registers are used in order to pipeline the rotators and allow
for a higher clock frequency.

Post place-and-route results for the Virtex-7 XC7VX330T
FPGA (V7 in the table) are shown in Table IV. The proposed
architectures achieve high clock frequencies, use a small
number of slices and they do not need any BRAM or DSP
block. For comparison to previous 1024-point FFTs, Table IV
also presents results on a Virtex-4 XC4VSX55 (V4 in the
These results agree with the conclusions from Table III: Compared to [5], the proposed design reduces the number of slices due to the lower number of adders and multipliers. Compared to [4] the proposed design uses more slices but removes the needs for BRAMs and DSP blocks. Finally, the proposed architectures achieve higher clock frequency compared to previous architectures.

Table V shows experimental results on ASICs. The proposed design shows good figures of merit in terms of clock frequency, area, power consumption, energy and SQNR, where the proposed approach is superior in most parameter. Furthermore, the area and power consumption of the proposed design could be further reduced by removing the pipelining used to increase the clock frequency. Normalized area and energy are calculated as:

\[
\text{Norm. Area} = \frac{\text{Area}}{\left(\frac{\text{Tech.}}{65 \text{ nm}}\right)^2}
\]

\[
\text{Energy (J/sample)} = \frac{\text{Power Consumption}}{\left(\frac{\text{Tech.}}{65 \text{ nm}}\right) \times f_{\text{CLK}} \times N}
\]
In order to quantify the improvement with respect to a usual FFT, the radix-$2^4$ 1024-point FFT has been compared to a radix-$2^4$ 1024-point FFT that uses multipliers instead of the proposed rotators. On FPGAs the latter requires 2235 $\times$ 24 % smaller in the proposed FFT. Finally, regarding accuracy for the proposed design. Power consumption and energy are 629 MHz. This is 34 % less area and 57 % more frequency

### VI. CONCLUSIONS

This paper shows how to design multiplierless unity-gain SDF FFT architectures. The proposed architectures are not only multiplierless and achieve unity gain, but also require the smallest number of adders among current SDF FFTs.

The proposed architectures achieve good figures of merit in terms of clock frequency, area, power consumption, energy, SQNR and FN.

### VII. ACKNOWLEDGMENT

The authors would like to thank Petter Källström for his help with the experimental results.

### REFERENCES


### TABLE IV

<table>
<thead>
<tr>
<th>Proposed Architecture</th>
<th>Radix-$2^4$</th>
<th>[7]</th>
<th>[10]</th>
<th>[8]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT Size</td>
<td>1024</td>
<td>64</td>
<td>512</td>
<td>2048</td>
</tr>
<tr>
<td>Radix</td>
<td>$2^4$</td>
<td>$2^4$</td>
<td>$2^4$</td>
<td>$2^4$</td>
</tr>
<tr>
<td>Word length</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Technology (nm)</td>
<td>65</td>
<td>65</td>
<td>180</td>
<td>90</td>
</tr>
<tr>
<td>Voltage (V)</td>
<td>1.1</td>
<td>1.1</td>
<td>-</td>
<td>1.2</td>
</tr>
<tr>
<td>Clk (MHz)</td>
<td>990</td>
<td>629</td>
<td>166</td>
<td>310</td>
</tr>
<tr>
<td>Latency (ns)</td>
<td>1.03</td>
<td>1.63</td>
<td>0.38</td>
<td>1.65</td>
</tr>
<tr>
<td>Area (mm$^2$)</td>
<td>0.28</td>
<td>0.37</td>
<td>0.47</td>
<td>0.78</td>
</tr>
<tr>
<td>Norm. Area</td>
<td>0.28</td>
<td>0.37</td>
<td>0.06</td>
<td>0.40</td>
</tr>
<tr>
<td>Comb. gates</td>
<td>870</td>
<td>1636</td>
<td>4139</td>
<td>2580</td>
</tr>
<tr>
<td>DSP Blocks</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Energy (pJ/sample)</td>
<td>-112.9</td>
<td>-114.26</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

††: Measured at 50 MHz.
††‡: Measured at 310 MHz.