Multiplierless Unity-Gain SDF FFTs

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

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, CCSSI**

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.

M. Garrido, R. Andersson and O. Gustafsson are with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, e-mails: mario.garrido.galvez@liu.se, rikan842@student.liu.se, oscar.gustafsson@liu.se

F. Qureshi is with the Department of Pervasive Computing, Tampere University of Technology, 33101 Tampere, Finland, e-mail: fahad@tut.fi

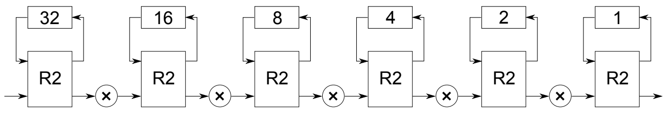

Fig. 1. A 64-point SDF FFT architecture, using $n = 6$ stages.

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 presents 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 presents 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

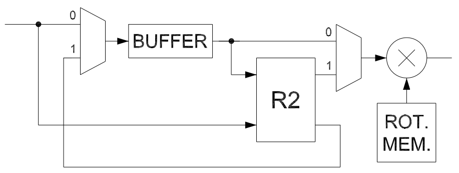

Fig. 2. Internal structure of a SDF stage.

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

	FFT Stage				
Radix	1	2	3	4	5
Radix-2	W_{64}	W_{32}	W_{16}	W_8	W_4
Radix-2^2	W_4	W_{64}	W_4	W_{16}	W_4
Radix-2^3	W_4	W_8	W_{64}	W_4	W_8
Radix-2^4	W_4	W_{64}	W_4	W_{16}	W_4

$W_L^\phi = e^{-j2\pi\phi/L}$. 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 ϕ, 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°, 90°, 180° and 270°). 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,

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

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

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

where N_C is the number of stages where CORDIC rotators are used, and the product over $s^\star$ 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 architectures. The first column shows the FFT algorithm. 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 $T_4M_{16}T_4C_{64}T_4$ 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 size of the rotator, its coefficients, scaling (R_s), rotation error (ϵ) and number of adders of the rotator (Add.), as defined in [11]. The second last column shows the number of adders of the entire FFT architecture. The total number of real adders for the FFT is calculated as

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

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 ± 2^{-9}, which represents an angle of $\alpha = 0.11°$. This resolution is enough for up to 1024-point FFTs, where the minimum rotation angle is $360°/1024 = 0.35°$. 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:

$$G_{FFT} = \frac{R_{tot}}{2^{\text{round}(\log_2 R_{tot})}} \tag{4}$$

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-2^3 512-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 ϵ 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 ($\star$).

V. Implementation

The two designs marked with a star ($\star$) 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.

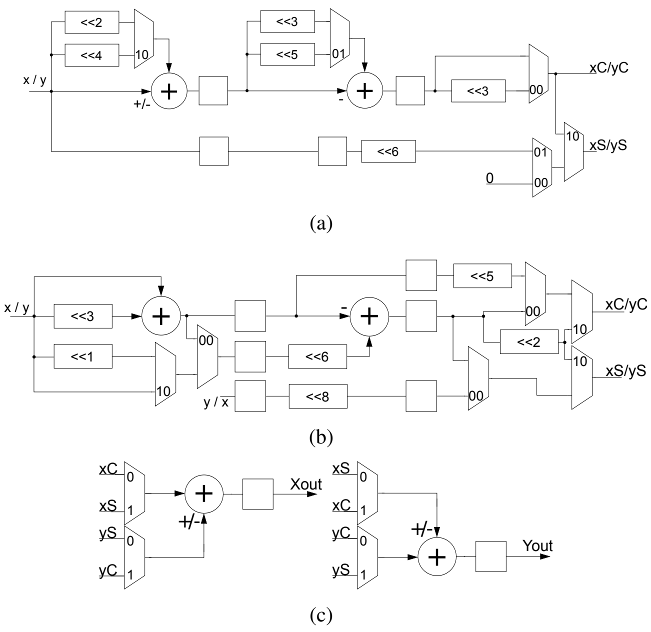

Fig. 3. Internal structure of the CCSSI W_{16} rotators in the implemented SDF FFTs. (a) Calculation of the products for the rotator $\{168,\ 155+j64,\ 119+j119\}$ in the 256-point SDF FFT. (b) Calculation of the products for the rotator $\{311,\ 288+j119,\ 220+j220\}$ in the 1024-point SDF FFT. (c) 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

TABLE II
PROPOSED UNITY-GAIN SDF FFT ARCHITECTURES.

FFT Algorithm (Twiddle Factors and Radix)	W_8, W_{16} and W_{32} Rotators: Size	Coefficients	R_s	ϵ	Add.	FFT Adders	G_{FFT}
64-point							
$T_4M_8C_{64}T_4M_8$ radix-2^3	W_8	$24, 17+j17$	24.02	$8.67 \cdot 10^{-4}$	6	50	1.0140
	W_8	$297, 210+j210$	296.99	$2.55 \cdot 10^{-5}$	6		
$T_4M_{16}T_4C_{64}T_4$ radix-2^4	W_{16}	$55, 51+j21, 39+j39$	55.12	$2.18 \cdot 10^{-3}$	8	56	1.0028
	W_{16}	$55, 51+j21, 39+j39$	55.12	$2.18 \cdot 10^{-3}$	8		
128-point							
$T_4M_{16}T_4C_{128}T_4M_8$ radix-2^4	W_{16}	$31, 29+j12, 22+j22$	31.19	$6.17 \cdot 10^{-3}$	6	60	0.9982
	W_8	$7, 5+j5$	7.04	$5.05 \cdot 10^{-3}$	6		
256-point							
$T_4M_{16}T_4C_{256}T_4M_{16}T_4$ radix-2^4	W_{16}	$24, 22+j9, 17+j17$	23.89	$6.53 \cdot 10^{-3}$	6	64	1.0015
	W_{16}	$37, 34+j14, 26+j26$	36.87	$3.47 \cdot 10^{-3}$	6		
$T_4M_{16}T_4C_{256}T_4M_{16}T_4$ radix-2^4	W_{16}	$24, 22+j9, 17+j17$	23.89	$6.53 \cdot 10^{-3}$	6	64	0.9987
	W_{16}	$147, 136+j56, 104+j104$	147.08	$2.09 \cdot 10^{-3}$	6		
$T_4M_{16}T_4C_{256}T_4M_{16}T_4$ radix-2^4	W_{16}	$168, 155+j64, 119+j119$	167.96	$1.96 \cdot 10^{-3}$	6	64	1.0025 $\star$
	W_{16}	$168, 155+j64, 119+j119$	167.96	$1.96 \cdot 10^{-3}$	6		
512-point							
$T_4M_8C_{512}T_4M_8C_{64}T_4M_8$ radix-2^3	W_8	$297, 210+j210$	296.99	$2.55 \cdot 10^{-5}$	6	94	0.9979
	W_8	$297, 210+j210$	296.99	$2.55 \cdot 10^{-5}$	6		
	W_8	$140, 99+j99$	140.01	$2.55 \cdot 10^{-5}$	6		
$T_4M_{16}T_4C_{512}T_4M_8M_{32}T_4$ radix-2^4	W_{16}	$102, 94+j39, 72+j72$	101.88	$1.21 \cdot 10^{-3}$	8	80	0.9994
	W_8	$338, 239+j239$	337.00	$4.38 \cdot 10^{-6}$	8		
	W_{32}	$209, 193+j80, 205+j41, \ldots$ $\ldots, 174+j116, 148+j148$	209.09	$1.06 \cdot 10^{-3}$	8		
1024-point							
$T_4M_{16}T_4C_{1024}T_4M_{16}T_4C_{64}T_4$ radix-2^4	W_{16}	$311, 288+j119, 220+j220$	311.37	$1.18 \cdot 10^{-3}$	6	92	1.0029 $\star$
	W_{16}	$311, 288+j119, 220+j220$	311.37	$1.18 \cdot 10^{-3}$	6		
$T_4M_{16}T_4C_{1024}T_4M_{16}T_4C_{64}T_4$ radix-2^4	W_{16}	$7, 7+j3, 5+j5$	7.31	$4.30 \cdot 10^{-2}$	6	92	0.9968
	W_{16}	$13, 12+j5, 9+j9$	12.87	$1.07 \cdot 10^{-2}$	6		
$T_4M_{16}T_4C_{1024}T_4M_{16}T_4C_{64}T_4$ radix-2^4	W_{16}	$6, 5+j2, 4+j4$	5.69	$5.47 \cdot 10^{-2}$	6	96	1.0022
	W_{16}	$133, 123+j51, 94+j94$	133.05	$8.58 \cdot 10^{-4}$	10		
$T_4M_8M_{32}T_4C_{1024}T_4M_8M_{32}T_4$ radix-2^5	W_8	$99, 70+j70$	98.00	$2.55 \cdot 10^{-5}$	6	84	0.9991
	W_{32}	$10, 9+j2, 8+j5, 9+j4, 7+j7$	9.58	$4.37 \cdot 10^{-2}$	6		
	W_8	$99, 70+j70$	98.00	$2.55 \cdot 10^{-5}$	6		
	W_{32}	$10, 9+j2, 8+j5, 9+j4, 7+j7$	9.58	$4.37 \cdot 10^{-2}$	6		

TABLE III
COMPARISON OF N-POINT SERIAL FFT ARCHITECTURES AND PARTICULARIZATION FOR $N = 1024$.

PIPELINED ARCHITECTURE		AREA (Any N)					AREA ($N = 1024$)	
		Butterflies	Rotators		Complex Memory		Real	Real
Type	Radix	Real Adders	Real Adders	Real Multipliers	Rotations	Data	Adders	Multipliers
SDF	Radix-2^2, [1]	$2(\log_2 N)$	$2(\log_4 N - 1)$	$4(\log_4 N - 1)$	N	$4N/3$	28 (108*)	16 (0*)
SDF	Radix-2^4, [2]	$4(\log_2 N)$	$2(\log_4 N - 1)$	$4(\log_4 N - 1)$	N	N	48 (128*)	16 (0*)
SDF	Radix-2^2, [3], [4]	$4(\log_2 N)$	$5(\log_4 N - 1)$	$3(\log_4 N - 1)$	N	N	60 (120*)	12 (0*)
SDF	Radix-2, [5]	$4(\log_2 N)$	$2(U + S)(\log_2 N - 2)$	0	N	N	232	0
SDF	Radix-2^2, [6]	$4(\log_2 N)$	$30(\log_4 N - 1)$ †	0	$N/4$	N	160†	0
SDF	Proposed, radix-2^4	$4(\log_2 N)$	See equation (3)	0	N††	N	92	0
SDF	Proposed, radix-2^5	$4(\log_2 N)$	See equation (3)	0	N††	N	84	0

$\star$: Number of adders when using CORDIC rotators instead of complex multipliers.
† : This architecture also uses $16(\log_4 N - 1)$ barrel shifters for the rotators, i.e., 64 barrel shifters for $N = 1024$.
†† : The rotation memory can be further reduced to 0 by using the angle generator in [16], instead of a rotation memory.

table). 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:

$$\text{Norm. Area} = \frac{\text{Area}}{(\text{Tech.}/65\text{ nm})^2} \tag{5}$$

$$\text{Energy (J/sample)} = \frac{\text{Power Consumption}}{(\text{Tech.}/65\text{ nm}) \times f_{CLK} \times N} \tag{6}$$

TABLE IV
COMPARISON OF SDF FFTS IMPLEMENTED ON FPGAS.

	Proposed			[4]	[5]	[3]
FFT Size	256	1024	1024	1024	1024	1024
Radix	2^4	2^4	2^4	2^2	2^2	2^2
Word length	16	16	16	16	n/a	16
Device	V7	V7	V4	V4	V4	S3E
Slices	870	1636	4139	2256	6936†	2580
Clk (MHz)	379	520	301	236	198	93
BRAMs	0	0	0	8	n/a	2
DSP Blocks	0	0	0	16	0	n/a

† : Approximated by dividing the number of LUTs by 4.

TABLE V
COMPARISON OF FFTS IMPLEMENTED ON ASICS.

	Proposed	Radix-2^4	[7]	[10]	[8]
FFT Size	1024	1024	64	512	2048
Radix	2^4	2^4	2^3	2^5	2^2
Architecture	SDF	SDF	SDF	MDF	SDF
Word length	16	16	16	12	10
Technology (nm)	65	65	180	90	350
Voltage (V)	1.1	1.1	-	1.2	2.7
Clk (MHz)	990	629	166	310	76
Latency (μs)	1.03	1.63	0.38	1.65	26.95
Area (mm^2)	0.28	0.37	0.47	0.78	7.58
Norm. Area	0.28	0.37	0.06	0.40	0.26
Comb. gates	11033	18649	-	-	45900
Sequential gates	30369	38514	-	-	78300
Power (mW)†	17.6	21.9	29.7	92.8††	526
Energy (pJ/sample)	0.34	0.42	3.35	0.42	0.95
SQNR (dB)	48.7	51.3	-	35	45.3
FN (dB)	-112.92	-114.26	-	-	-

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

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 slices and achieves a clock frequency of 385 MHz, i.e., 37 % more area than the proposed approach and 35 % less clock frequency. On ASICs the comparison is presented in Table V: The proposed FFT takes up 0.28 mm^2 at 990 MHz, whereas the radix-2^4 FFT that uses multipliers takes up 0.37 mm^2 at 629 MHz. This is 34 % less area and 57 % more frequency for the proposed design. Power consumption and energy are 24 % smaller in the proposed FFT. Finally, regarding accuracy the SQNR of the proposed approach is 48.7 dB and the use of multipliers leads to 51.3 dB. Thus, there is a drop of half a bit in SQNR in the proposed design, but it is still at a very high level. Likewise, the Frobenius Norm [12] of both approaches is very similar: FN $= -112.92$ dB for the proposed approach and FN $= -114.26$ dB for the use of multipliers.

VI. CONCLUSIONS

This paper shows how to design multiplierless unity-gain SDF FFT architectures. The proposed architectures are not only multipierless 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

[1] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 7, pp. 585–589, Jul. 2006.

[2] A. Cortés, I. Vélez, and J. F. Sevillano, "Radix r^k FFTs: Matricial representation and SDC/SDF pipeline implementation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

[3] G. Zhong, H. Zheng, Z. Jin, D. Chen, and Z. Pang, "1024-point pipeline FFT processor with pointer FIFOs based on FPGA," in *Proc. IEEE Int. Conf. VLSI-Soc*, Oct. 2011, pp. 122–125.

[4] B. Zhou, Y. Peng, and D. Hwang, "Pipeline FFT architectures optimized for FPGAs," *Int. J. Reconf. Comp.*, Jun. 2009.

[5] S. Abdullah, H. Nam, M. McDermot, and J. Abraham, "A high throughput FFT processor with no multipliers," in *Proc. IEEE Int. Conf. Comput. Design*, Oct 2009, pp. 485–490.

[6] N. H. Cuong, N. T. Lam, and N. D. Minh, "Multiplier-less based architecture for variable-length FFT hardware implementation," in *Proc. IEEE Int. Conf. Comm. Electron.*, Aug. 2012, pp. 489–494.

[7] T. Adiono, M. S. Irsyadi, Y. S. Hidayat, and A. Irawan, "'64-point fast efficient FFT architecture using radix-2^3 single path delay feedback'," *Electrical Engineering and Informatics, 2009. ICEEI '09. International Conference on*, vol. 02, pp. 654 – 658, Aug. 2009.

[8] T. Lenart and V. Öwall, "Architectures for dynamic data scaling in 2/4/8k pipeline FFT cores," *IEEE Trans. VLSI Syst.*, vol. 14, no. 11, pp. 1286–1290, Nov 2006.

[9] M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. VLSI Syst.*, vol. 21, pp. 23–32, Jan. 2013.

[10] T. Cho and H. Lee, "A high-speed low-complexity modified radix-2^5 FFT processor for high rate WPAN applications," *IEEE Trans. VLSI Syst.*, vol. 21, no. 1, pp. 187–191, Jan. 2013.

[11] M. Garrido, F. Qureshi, and O. Gustafsson, "Low-complexity multiplierless constant rotators based on combined coefficient selection and shift-and-add implementation (CCSSI)," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 7, pp. 2002–2012, Jul. 2014.

[12] M. Garrido, O. Gustafsson, and J. Grajal, "Accurate rotations based on coefficient scaling," *IEEE Trans. Circuits Syst. II*, vol. 58, no. 10, pp. 662–666, Oct. 2011.

[13] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computing*, vol. EC-8, pp. 330–334, Sep. 1959.

[14] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.

[15] C.-S. Wu, A.-Y. Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Trans. Circuits Syst. II*, vol. 50, no. 9, pp. 589–601, Sep. 2003.

[16] M. Garrido and J. Grajal, "Efficient memoryless CORDIC for FFT computation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 2, Apr. 2007, pp. 113–116.

[17] W. Han, A. T. Erdogan, T. Arslan, and M. Hasan, "High-performance low-power FFT cores," *ETRI J.*, vol. 30, no. 3, pp. 451–460, Jun. 2008.

[18] C.-H. Yang, T.-H. Yu, and D. Markovic, "Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 757–768, Mar. 2012.

[19] P. Källström, M. Garrido, and O. Gustafsson, "Low-complexity rotators for the FFT using base-3 signed stages," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, Dec. 2012, pp. 519–522.

[20] S. Oraintara, Y.-J. Chen, and T. Nguyen, "Integer fast fourier transform," *IEEE Trans. Signal Process.*, vol. 50, no. 3, pp. 607–618, Mar 2002.

[21] A. G. Dempster and M. D. Macleod, "Multiplication by two integers using the minimum number of adders," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, May 2005, pp. 1814–1817.

[22] O. Gustafsson, "A difference based adder graph heuristic for multiple constant multiplication problems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1097–1100.

[23] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, pp. 1–39, May 2007.

[24] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, 1965.

[25] F. Qureshi and O. Gustafsson, "Generation of all radix-2 fast Fourier transform algorithms using binary trees," in *Proc. European Conf. Circuit Theory Design*, Aug. 2011, pp. 677–680.