Exploring the CORDIC Algorithm and Clock-Gating for Power-Efficient Fast Fourier Transform Hardware Architectures

André Sapper[1], Guilherme Paim[2], Eduardo A. C. da Costa[1], Sergio Bampi[2]

[1]Graduate Program in Electronic Engineering and Computing - Catholic University of Pelotas (UCPel), Pelotas - Brazil
[2]Graduate Program in Microelectronics (PGMicro) - Federal University of Rio Grande do Sul (UFRGS), Porto Alegre - Brazil
e-mail: {gppaim, bampi}@inf.ufrgs.br, {eduardo.costa}@ucpel.edu.br

***Abstract*— This work explores hardware-oriented optimizations for the CORDIC (COordinate Rotation Digital Computer) algorithm investigating the power-efficiency improvements employing N-point Fast Fourier Transform (FFT) hardware architectures. We introduced three hardware-oriented optimizations for the CORDIC: (a) improving the sign extension, (b) removing the angle accumulation and (c) eliminating the redundancies in the iterations, both unnecessary when processing the FFT. Fully sequential FFT architectures of 32, 64, 128, and 256 points were synthesized employing ST 65 nm standard cell libraries. The results show up to 38% of power savings on average when using our best CORDIC optimization proposal to the FFT architecture comparing to the explicit multiply-based butterfly version. Moreover, when combining our best CORDIC optimization with the clock-gating technique, the power savings rises to 78.5% on average for N-point FFT.**

***Index Terms*— Fast Fourier Transform, Low-power, CORDIC, Clock-gating, Hardware Architectures.**

I. Introduction

The internet of things (IoT) era has become real with a wide range of consumer electronics. On the other hand, the foundries are announcing the end of the Moore's law [1] technology scaling in the next-generation nanometric technologies. The most popular portable equipment, especially those operated by a battery like smartphones, smartwatches, and notebooks (among many others), have both constraints as high-performance and autonomy of higher utilization. In practice, each new embedded electronic device is launched with less weight, less thickness, and faster than its previous version, but without significant improvements in the duration of the battery.

Digital signal processing (DSP) algorithms are widely embedded in battery-constrained IoT. In DSP applications, one of the most commonly used operations involves the processing of the signal by the discrete Fourier transform (DFT). This class of algorithm is used in such areas as image processing, audio processing, biomedical engineering, communications, among others. The work presented in [2] summarizes in a brief overview of the significant developments in FFT algorithms along with some popular applications in speech and image processing, signal analysis, and communication systems.

In the mentioned application areas, the FFT algorithms process the DFT efficiently by saving arithmetic operations. The butterfly represents the central element in the FFT algorithm, whose samples are multiplied by twiddle factors. The twiddle factors represent multiple values of N = 2, where N is the number of FFT points. One of the methods employed to generate twiddle factors is the CORDIC algorithm [3]. The key idea of using CORDIC in FFT is to replace cosine and sine factors by iterative rotations. It allows reducing the requirements of registers or read-only memory (ROM), as well as replacing the multiplication operations with simpler operations of shift and additions.

This paper explores the use of the CORDIC algorithm in FFT architectures. We explore implementations of the unfolded CORDIC algorithm with and without the use of the pipeline. Therefore, we can present four different versions of the CORDIC core: (i) using the traditional algorithm for cosine and sine representations, (ii) exploring the redundancies of the sign extension, (iii) eliminating the angle accumulator, and (iv) combining both (ii) and (iii).

Fully-sequential FFT architectures were implemented as a case study, with the CORDIC embracing also the twiddle factors generation. According to our synthesis results, employing our best CORDIC optimization proposal in the FFT architecture results in up to 38% of power savings on average for N-point FFT. The sequential structure is fully parameterized, and it uses shift-registers, which enables the use of the clock-gating technique. When combining our best CORDIC optimization with the clock-gating technique, the power savings rises to 78.5% on average for N-point FFT.

Our work investigates the power-efficiency improvements when employing the following optimizations into the FFT architecture:
1) Baseline: implementing circular shift-registers in an explicit butterfly (employing multipliers), i.e., storing all the values of the parts real and imaginary coefficients.
2) CORDIC: employing an optimized CORDIC herein proposed (storing the angles of the twiddle factors).
3) Clock-gating: employing clock-gating in the shift-registers.
4) CORDIC and clock-gating: combining clock-gating in the shift registers and the optimized CORDIC proposal in both the twiddle factors and the butterfly operations.

The main results showed that the employing of our optimized CORDIC in the FFT architecture based on the clock-gating technique promotes notable power dissipation savings independent for 32, 64, 128 and 256-point FFT architectures synthesized for ST 65 nm standard cells.

The **key contributions** of this work are as follows:
1) Exploring architectural optimizations for a power-efficient CORDIC hardware design.
2) Investigating the power-efficiency improvements on the N-point FFT architectures when employing our CORDIC

Digital Object Identifier 10.29292/jics.v16i2.226

architecture optimization proposals as well as combining the best proposal with clock-gating.

This paper is structured into five sections. Section II presents the background of concepts of both FFT and the CORDIC algorithm. Section III gives details of dedicated hardware implementations of the CORDIC algorithm. Section IV shows aspects of implementing a sequential FFT architecture and the use of the CORDIC algorithm for twiddle factors generation. Section V shows our investigation results for our CORDIC optimization employed in the clock-gated FFT architectures. Finally, Section VI concludes the work.

II. Background

This section summarizes the main aspects of the FFT algorithm, as well as the CORDIC algorithm. This section also shows an overview of significant works from the literature.

A. Fast Fourier Transform overview

The FFT is an efficient algorithm to calculate the DFT. The DFT is defined as a discrete sequence in the frequency domain, and its exponential format is given in (1). The simplified form of the DFT is shown in (2), where W_N represents the twiddle factors, or trigonometric coefficients, and m is the *m-th* DFT output component, where $W_N = e^{-j2\pi/N}$ and $m = 0, 1, 2, 3, .., N-1$.

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nm/N} \tag{1}$$

$$X(m) = \sum_{n=0}^{N-1} x(n) W_N \tag{2}$$

The amount of required calculation of the DFT is $O(N^2)$. One uses the FFT algorithm to reduce a large amount of calculation needed by the DFT. The work in [4] introduced the most popular FFT algorithm. The main idea is recursively processing $N/2$ even and odd samples, where the number of calculations is reduced to $O(NlogN)$. Fig. 1 shows an example of an N = 8 points FFT flow with decimation in time (DIT). Note that, for this case, the input samples (first stage) are divided (decimated) in even and odd. It repeats for all remaining stages.

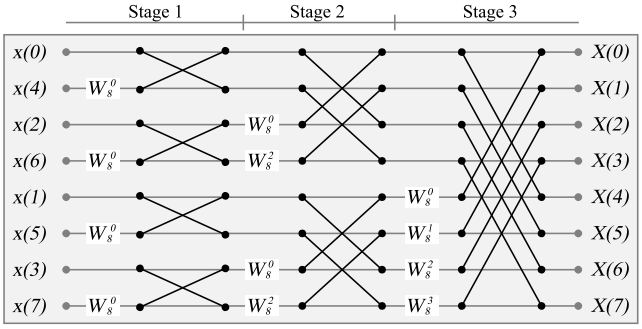

Fig. 1: Example of the flow of an N=8-point DIT FFT.

The main element in the FFT algorithm is called butterfly. The butterfly plays a central role in the FFT computation. This operator performs the calculation of complex terms, which involves multiplication of input data by appropriate coefficients [4]. Fig. 2 shows the architecture of the radix-2 multiply-based butterfly for DIT operations. In Fig. 2, A, and B are complex inputs, W is the complex constant (twiddle factor), and C and D are the complex outputs.

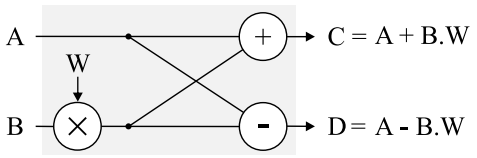

Fig. 2: Structure of the radix-2 DIT multiply-based butterfly.

B. CORDIC algorithm overview

CORDIC algorithm solves [5], in a computing operation and with equal speed: (i) the relationships involved in the rotation of coordinates in the plane; (ii) conversion of rectangular to polar coordinates; (iii) multiplication; (iv) division; or (v) the conversion between base systems. Also, a variety of functions can be derived from these functions using vector rotations, as shown in Table I. CORDIC's application areas expanded to address various DSP algorithm problems such as filtering, equalization, FFT, and image processing. The efficient implementation of CORDIC is a crucial point for the practical realization of dedicated systems. CORDIC provides a method iterative to perform vector rotations by arbitrary angles using only shifts and additions. Due to the simplicity of operations, it is suitable for VLSI (very large scale integration) design.

Table I.: CORDIC: Operation modes.

		Modes	
Coord.	Out	Rotation	Vectorization
Circ. $m = 1$	x	$K(x\cos z - y\sin z)$	$K\sqrt{x^2+y^2}$
	y	$K(y\cos z + x\sin z)$	0
	z	0	$z + \arctan(y/x)$
Lin. $m = 0$	x	x	x
	y	$y + xz$	0
	z	0	$z + y/x$
Hyp. $m = -1$	x	$K'(xcoshz - ysinhz)$	$K\sqrt{x^2+y^2}$
	y	$K'(ycoshz + xsinhz)$	0
	z	0	$z + \tanh^{-1}(y/x)$

K≈1.647

Fig. 3 shows a concrete example of the CORDIC operation with the values of each iteration. The inputs in X, Y and Z are, respectively, $\sqrt{2}/2$ (0.70711), $\sqrt{2}/2$, and $\pi/4$, i.e., we want to rotate a vector with 45 degrees of phase by 45 degrees. As the CORDIC is an iterative algorithm as iterations occur closer to the desired result we are.

C. Related work

According to [6], the most common way of implementing an FFT is the use of a pipelined technique for increasing throughput of architecture. The work presented here follows this trend with the proposition of a fully-sequential architecture using the pipeline technique.

The work in [6] presents a comparison between the different structure's implementation of FFT architectures. The

```
>> [COS,SIN]=cordic(0.70711,0.70711,pi/4,9,9,0)
```

	X - (y * d)		Y + (x * d)		d(+>0; -<0) Z - Z_tep	
INPUTS	X: 0010110101	0.7071	Y: 0010110101	0.7071	Z: 0011001001	0.7854
	-x: 0010110101	0.7071	+y: 0010110101	0.7071	z: 0011001001	0.7854
IT: 1	X: 0000000000	0.0000	Y: 0101101010	1.4142	Z: 0000000000	-0.0000
	+x: 0010110101	0.7071	+y: 0000000000	0.0000	z: 0001110111	0.4636
IT: 2	X: 0010110101	0.7071	Y: 0101101010	1.4142	Z: 0001110111	0.4636
	-x: 0001011011	0.3536	+y: 0000101101	0.1768	z: 0000111111	0.2450
IT: 3	X: 0001011011	0.3536	Y: 0110010111	1.5910	Z: 0000111000	0.2187
	-x: 0000110011	0.1989	+y: 0000001011	0.0442	z: 0000100000	0.1244
IT: 4	X: 0000101000	0.1547	Y: 0110100011	1.6352	Z: 0000011000	0.0943
	-x: 0000011010	0.1022	+y: 0000000010	0.0097	z: 0000010000	0.0624
IT: 5	X: 0000001101	0.0525	Y: 0110100101	1.6449	Z: 0000001000	0.0319
	-x: 0000001101	0.0514	+y: 0000000000	0.0016	z: 0000001000	0.0312
IT: 6	X: 0000000000	0.0011	Y: 0110100110	1.6465	Z: 0000000000	0.0007
	-x: 0000000111	0.0257	+y: 0000000000	0.0000	z: 0000000100	0.0156
IT: 7	X: 1111111010	-0.0246	Y: 0110100110	1.6465	Z: 1111111100	-0.0150
	+x: 0000000011	0.0129	+y: 0000000000	-0.0002	z: 0000000010	0.0078
IT: 8	X: 1111111101	-0.0118	Y: 0110100110	1.6467	Z: 1111111110	-0.0072
	+x: 0000000010	0.0064	+y: 0000000000	-0.0000	z: 0000000001	0.0039
IT: 9	X: 1111111111	-0.0054	Y: 0110100110	1.6468	Z: 1111111111	-0.0032

```
COS = -0.0053517          SIN =  1.6468
```

Fig. 3: A CORDIC algorithm numerical example.

widely adopted architectures and trends in architectural modification to reduce power consumption and area and to achieve high throughput are presented in [7].

The work in [8], proposes a low-power, high-speed reconfigurable FFT processor. The design consists of one radix-2 processing element (PE), two radix-4 PE, and two radix-8 PE, which are put together in a pipeline single-path delay feedback (SDF) architecture. However, according to [6], although the authors claim to have proposed a low power project, the results do not show it since the architecture presents 307.7 mW of power dissipation. The work presented in [9] aims to achieve the ideal balance between low power and high flexibility. The system can be reconfigured to perform 16 to 1024-point FFTs using only one butterfly block. Meantime, the proposed solutions cannot be considered low power, since they present, on average, 68.7 mW and 81.8 mW for the non-reconfigurable and reconfigurable architectures, respectively.

The work in [10] shows an FFT architecture supporting both variable window sizes and output pruning. The FFT supports 128-2048/1536 point FFT, as required by long term evolution (LTE) systems. The work does not present power results. Radix-2 and radix-2^k feedforward architectures based on rotator allocation are presented in [11]. The rotator allocation approach consists in distributing the rotations of the FFT. The main goal is twofold reduces the number of edges in the FFT with rotators and reduces the complexity of the rotators. The work in [12] proposes a high-speed and area-efficient 64-point FFT processor using the Vedic algorithm. The work in [13] presents an architecture using a reconfigurable structure with constant multipliers and parallel multipliers of bits (using Booth's multiplication algorithm) to generate the twiddle factors. According to the authors, this reduces the cost of hardware compared to use a large ROM memory. The work in [14] uses the simple concept of hashing and lookup table to effectively reduce the number of arithmetic operations required to perform the FFT of an electrocardiogram (ECG) signal.

Other works explore some techniques to reduce memory size ROM used, as in [15] that, to reduce the area of the circuit, uses canonical signed digit (CSD) notation and a non-multiplier unit that does not stores all twiddle factors in ROM. Only a few twiddle factors are stored, and the rest can be derived using only shift and sum operations. Also, the authors took into account that the inputs will only be real to reduce energy consumption. In this way, the first stage of butterflies is modified to ignore imaginary information. However, it can be a problem for some target applications that also require imaginary values. A serial commutator (SC) FFT was proposed in [16]. The proposed SC FFT uses circuits for bit-dimension permutation of serial data. As a result, the proposed architectures use the minimum number of adders, rotators, and memory that are necessary for a pipelined FFT of serial data.

The work presented in [17] uses complex multiplication for the generation of twiddle factors. These twiddle factors are half the number of FFT points, and based on the stage used, the number of coefficients required also varies. The proposed architecture was synthesized in CMOS 0.18 um, and works with 36 bits input size, at 100 MHz, and consumes 39 mW, at a supply voltage of 1.8 V. An embedded design of a 1024-point FFT processor architecture for high-performance harmonic measurement techniques are proposed in [18]. The FFT processor algorithm incorporates both pipelining and parallel strategies to enhance performance. The proposed FFT makes use of the floating-point format to realize higher precision FFT.

From the mentioned works until now, only [8] employs the CORDIC algorithm for the generation of twiddle factors. However, as shown in the literature, as well as in this work,

Table II.: Related Work on the FFT Design Optimization.

Related Work	A	B	C	D	E	F
[6, 11]	FPGA/ASIC	✓	×	×	×	×
[8]	ASIC	✓	×	✓	×	✓
[9]	FPGA/ASIC	✓	×	×	×	✓
[10]	FPGA	×	×	×	×	×
[12]	FPGA	✓	×	×	×	×
[13]	ASIC	×	×	×	×	×
[14–17]	ASIC	✓	×	×	×	×
[18]	FPGA/ASIC	✓	✓	×	×	×
[20]	FPGA/ASIC	✓	×	✓	×	×
[21, 22]	N/A	×	×	✓	×	×
[24, 25]	FPGA	✓	×	✓	×	×
[26]	N/A	×	×	✓	×	×
[27–30]	FPGA	×	×	✓	×	×
Our work	ASIC	✓	✓	✓	✓	✓

(A) Target Device for Results, **(B)** Power Results,
(C) Methodology for Power Analysis,
(D) Employing CORDIC in the FFT,
(E) Exploring hardware-oriented CORDIC optimizations,
(F) Employing both CORDIC and clock-gating.

the CORDIC algorithm is the most efficient way of generating twiddle factors. The work in [19] presents a survey on CORDIC based FFT implementation on FPGA.

The work in [20] proposes a low-power and high-speed CORDIC-based split-radix FFT processor for OFDM systems. A called CORDIC-friendly FFT architecture is proposed in [21]. The design technique showed in [21] reduces the hardware complexity compared to the direct implementation of the butterflies using complex multipliers. An extension of this work was presented in [22], where instead of optimizing each stage independently, one considers the joint optimization of two stages of the CORDIC-Friendly FFT rotations. The work in [23] proposes a modified CORDIC based FFT using Vedic multiplication techniques. However, neither of them present power dissipation results.

The work in [24] proposes a design of the FFT processor using a CORDIC algorithm to reduce the complexity of hardware and improve the performance of processing. Although the authors claim that the purpose of the work is to obtain an area-efficient description of the FFT processor, there are no comparisons to support it. The design of a radix-2 64-point FFT processor using the CORDIC algorithm is proposed in [30]. The proposed algorithm uses a new addressing scheme and the angle generator logic to remove the ROM usage to store twiddle factors. However, there are no power dissipation results.

The work in [26] proposes an FFT architecture based on the CORDIC algorithm for the generation of coefficients. The authors claim that the proposed architecture based on CORDIC reduces the use of memory resources widely. However, they do not present results of circuit area and power dissipation that can support such statement. The work in [27] discusses the realization of a radix-2 FFT processor based on the CORDIC algorithm on FPGA. However, power results are not available. The work in [28] proposes a binary FFT architecture based on the use of the CORDIC algorithm. According to the authors, this is the first architecture that computes output string values using the CORDIC algorithm. Although the authors state that the proposed solution reduces memory requirements as well as the amount of memory resources hardware, there are no results to support such a claim.

The design of a CORDIC algorithm based radix-4 FFT processor for spectrum analyzer using FPGA is proposed in [29], whose intention is its application in power harmonics signal processing. The work does not provide power results. The work presented in [25] proposes an N-point FFT architecture using the CORDIC algorithm. The CORDIC based on the Vedic multiplier is used on the butterfly to improve speed and the modified CORDIC processor. It optimizes area utilization in the calculation of twiddle factors. Although the authors present comparisons and gains with other works in the literature, the results presented are based only on the FPGA platform, which is not recommended for the evaluation of power dissipation values.

Although the works mentioned until now try their best to optimize the hardware features for an implementation of a low-power dissipation FFT, only one of them uses the clock-gating technique, together with the CORDIC algorithm. However, it does not explore the elimination of redundancies into the iterations in the CORDIC core, as we perform in this work. This aspect will be further explored, with possible optimizations proposals. Table II shows the main contributions of the literature as well as the contributions of our proposed solution.

III. CORDIC Architectural Optimizations

The proposed CORDIC architectures are implemented in two different architectural approaches using the same core, as can be seen in Fig. 4(a) and Fig. 4(b). Both architectures implement the CORDIC algorithm in extended form and the only difference between these two architectures is that Fig. 4(a) does not use pipeline stages, whereas Fig. 4(b) does.

The structure in Fig. 4(c) represents the common core of both CORDIC solutions and each core performs an iteration. Note that the core is only composed of additions/subtractions

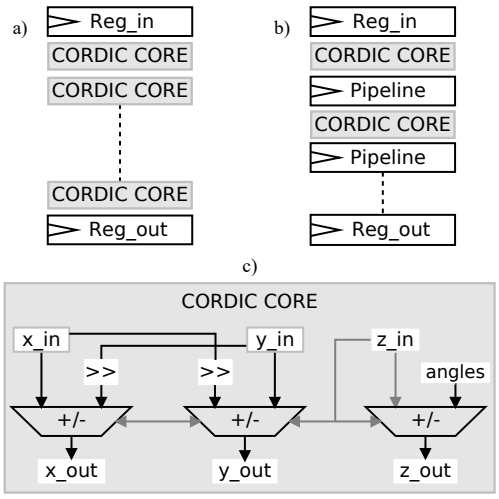

Fig. 4: CORDIC architecture: (a) fully-parallel, (b) fully-parallel with n-1 pipeline stages, where n is the number of iterations, (c) one CORDIC core used in Fig. 4 (a) and Fig. 4 (b).

and shifts, which is easy to implement in hardware. Therefore, combining a simple basic structure with the use of a pipeline tends to maximize the overall performance of the architecture. When implementing a pipeline version of CORDIC, one can achieve an increased throughput since each iteration occurs in each clock cycle. It is also advantageous for reducing energy dissipation as there is a significant reduction in glitching activity. The work in [31] explored both the combination of the number of bits of input data size and the number of iterations for a power-efficient fixed-point CORDIC implementation in addition to applying the pipeline technique. Therefore, our decision of choosing 16-bit fixed-point representation is based on the previous exploration of [31].

A. *Four versions of CORDIC*

Based on the implementation of Fig. 4, this work explores three more strategies for CORDIC architecture implementation, totaling four versions. Version 1 (v.1) is the base version of the CORDIC, while version 2 (v.2) and version 3 (v.3) implement different degrees of optimization in the base version. Version 4 (v.4) combines the features explored in v.2 and v.3.

Version 1 has the operation details shown in Fig. 5 which is the classic CORDIC algorithm, for cosine (cos) and sine (sin) representations, without any optimizations. Fig. 6 shows the operation details developed in versions 2 and 3 which combined form version 4.

Version 2 exploits the redundancy feature of the sign extension. It proceeds only where the intermediate value of the previous iteration was optimized was positive, i.e., where its most significant bit was z ero. Version 3 eliminates the z angle accumulator processing. In rotation mode, the angle accumulator z is initialized to the desired angle of rotation (see Table I), and its execution is done to reduce the residual magnitude of the angle ($z \rightarrow 0$). In the FFT, the values of the angles of rotation are previously known (twiddles) and can be precomputed, thus eliminating the accumulator of angles. This way, for a previously known set of input angles, the signals controls which are the signs of the z iterations (arrows in blue in Figs. 5 and 6) are stored in a ROM memory in place of the angles just as they would be.

Version 4 combines features of the solutions explored in CORDIC versions 2 and 3. Besides the accumulator angle has been eliminated, the technique also removes the

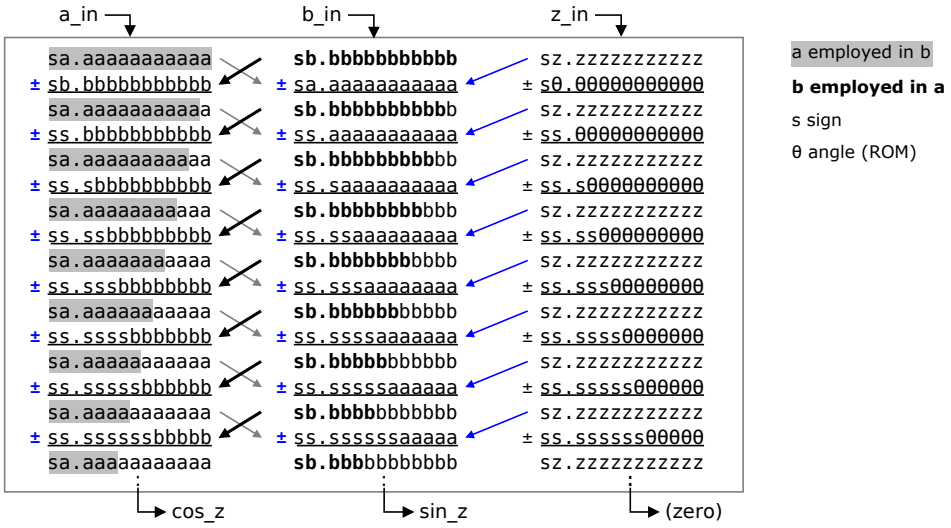

Fig. 5: CORDIC core version 1 (v.1).

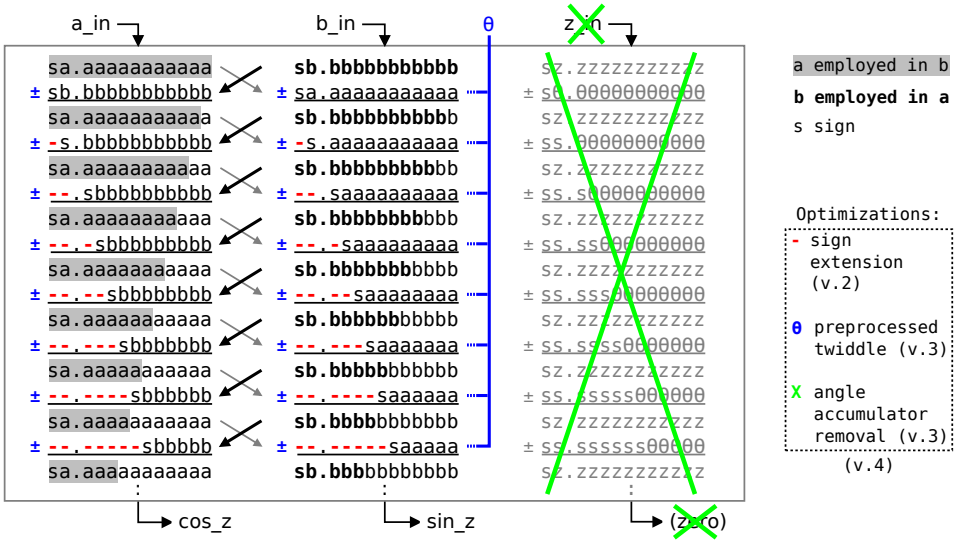

Fig. 6: CORDIC core version 4 (v.2 + v.3).

signal redundancy between iterations. The main difference between the different versions, essentially, is that versions 3 and 4 do not use the angle accumulator, which can be pre-processed when the angle of input is previously known, which is the case with the architectures implemented in this work. This feature in the implementation aims to guarantee a smaller area and a lower power dissipation.

IV. FFT ARCHITECTURES

Fig. 7 shows the structure of the FFT architecture proposal. The state machine (FSM) controls a total of six main multiplexers (muxes), organized as follows: the multiplexers `mux0`, `mux1` and `mux2`, select the operation mode (serial or parallel) of the data shift-registers (`sr0`, `sr1` and `sr2`). The multiplexers `mux3` and `mux4` select the inputs for processing by the butterfly. Specifically, the `mux3` selects between outputs `sr0` and `sr2` for the butterfly input `x0`. Equivalently, the `mux4` selects from `sr1` output and the `data_in_register` output to the butterfly input `x1`. The multiplexer `mux5` selects between `data_in_register` and butterfly output `y0` for `sr2` input.

The data memory is composed of three shift-registers: `sr0`, `sr1` and `sr2`. Each one of these shift-registers consists of two shift-registers in the parallel input, parallel output configuration. The main difference between them is that `sr2` has dual behavior. It is responsible for receiving $N/2$ as input samples; therefore, its input requires multiplexers. It is worth mentioning that the three shift-registers have an individual `clock_enable` control, allowing control optimizations.

The clock-gating technique is used in the three shift-registers, as highlighted in blue in Fig. 7. We design the clock-gating activating signal with a Johnson-type binary counter responsible for enabling and coordinating the individual switching of the data memory shift registers (`sr0`, `sr1` and `sr2`). The main purpose of this circuit is to provide individual control over the registers that make up the shift registers in the data memory via a `clock_enable` signal. The advantage of this control signal is that only registers that need to be updated are activated, which eliminates unnecessary switching providing dynamic power savings. The direct effect of this design technique is that the synthesis tool infers the generation clock-gating automatically from the enable clock signals.

Our FFT architecture employs the DIT radix-2 butterfly realization. The multiply-based butterfly version (i.e., without CORDIC) implements the arithmetic operators automatically inferred by the synthesis tool. The CORDIC-based butterflies versions are free from multiplications since they realize addition-and-shift operations.

The coefficients (i.e., twiddle factors) are constant values that need to be stored. Storing data in hardware requires some memory, and if the volume of the data is high, its cost can become prohibitive. Therefore, another alternative is the generation on the fly of twiddle factors when necessary, using, for example, CORDIC. The values of twiddle factors are, in practice, complex components (i.e., real and imaginary parts) that represent angles in their Cartesian form. These angles are symmetrical and mirrored throughout the unitary circle, i.e., they present redundancy that allows optimizations.

At first glance, the most straightforward implementation of twiddle factors is their storage in memory. Assuming an FFT of N points, it would take $N/2$ * 2 * n_{bits} bits of memory being $N/2$ the total of twiddle factors, two components (i.e., real and imaginary) of each twiddle, and n_{bits} of bit-width of each twiddle. For 128-point FFT and 16-bit resolution per twiddle component, we would then have $64 * 2 * 16$ totaling 2048 bits (2kB) of memory, not consid-

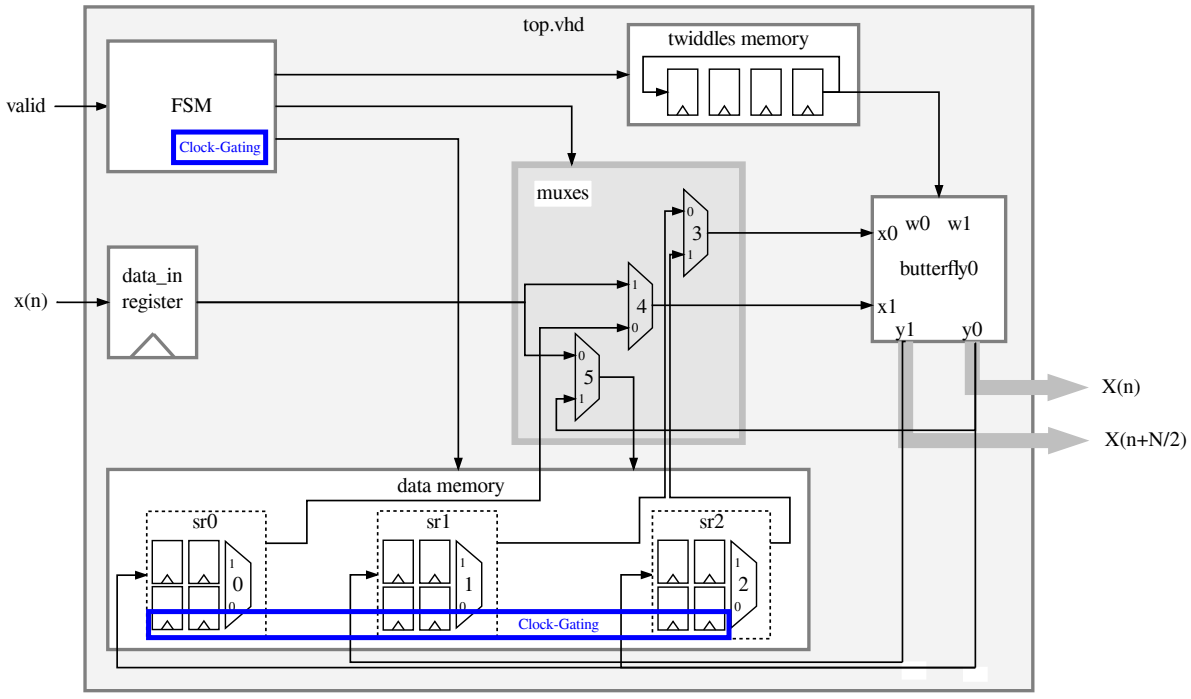

Fig. 7: Our FFT architecture employing clock-gating (highlighted in blue-colored).

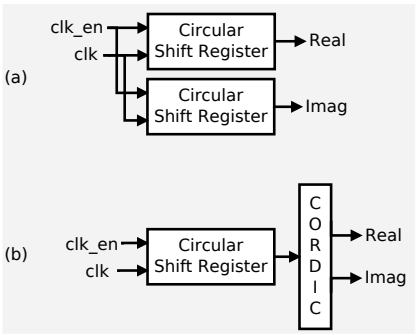

Fig. 8: Our FFT architecture employing (a) multiply-based butterfly and (b) CORDIC for the twiddle factors generation.

ering the hardware required for reading/writing. As noted, the requirements increase linearly with increasing FFT size, which for large hardware can be a costly consumer of resources.

We used two strategies for supplying trigonometric coefficients, as shown in items (a) and (b) of Fig. 8. Architecture (a) applies only a circular shift-register, with the sequences of each stage explicitly stored, that is, fully stores the values of the real parts and imaginary. Architecture (b) also uses a circular shift-register, and additionally, a CORDIC block. The shift-registers of this architecture stores only the twiddle angles factors of the sequences. It is to provide these angles to be processed by the CORDIC to obtain the real and imaginary values of each coefficient. The central idea of these two strategies is to evaluate the benefit of storing less information (only the angles trigonometric measurements) at the cost of using CORDIC.

V. Results and Discussions

This section presents the main results obtained in this work. Figs. 9 shows the results for the hardwired CORDIC optimizations. Fig. 10 shows the methodology for power dissipation analysis. Table III shows the total power results (in mW) for different parameters in a sequential architecture, for the FFT architecture proposals.

The architectures CORDIC and FFT are described in VHDL. A synthesis using Genus^{TM}tool in PLE^{TM}(physical layout estimation) mode generates the gate-level netlist and standard delay format (SDF) files. The Genus^{TM}receives: (i) the netlist, (ii) the ST 65 nm cell libraries at 1.0V, (iii) the constraint file (v) the Macro and tech library exchange format (LEF) files, and (v) the Table file of capacitance. The values of the input cosine and sine function are used and stored in text files. The simulation tool uses these values and runs the $testbench$ with the design files of the CORDIC architectures and generates a switching activity in a VCD format. VCD and the gate-level netlist are delivered for the synthesis tool that makes the circuit area, power dissipation, and delay reports. Power dissipation is estimated considering a source of 1.0 V.

A. Synthesis Results for the CORDIC Proposals

The syntheses varied the clock cycle period from 13.33 ns to 300 ns with five ns of increments. Therefore, these values ranged from 3.33 to 75 MHz, as presented in the curves of Figs. 9. As can be seen in the curves, the power increases as higher the frequency operation is. Fig. 9 shows that until close to 40 MHz, there is almost no difference in terms of power consumption between the CORDIC architectures. However, up to 40 MHz, versions 3 and 4 present less power consumption. Notably, version 4 is more power-efficient in the frequency close to 75 MHz. It occurs because version 4 explores both strategies of eliminating the accumulator angle and also eliminates redundancies between iterations. It is more significant for higher frequencies values (up to 60 MHz) because the impact of both strategies is more evident with the higher switching activity. Note that the use of both strategies is strongly beneficial for the area reduction in version 4 for frequencies operation up to 20 MHz, as can be seen in the curves of Fig. 9(a) and (d). It also explains the least power consumption presented by version 4.

We analyzed and present in Figs. 9(c) and 9(f) the circuit area behavior for each CORDIC version when raising the target clock frequency in the synthesis. The circuit area increases due to the synthesis effort to attend harder constraints imposed by higher target clock frequencies. Observing Figs. 9(c) and 9(f) it is concluded that versions 2 and 3 presents lower circuit area under the same target clock frequency. Furthermore, as shown in Figs. 9(b) and 9(e), static power consumption (i.e., leakage power) is lower in versions 3 and 4, version 4 being the smallest. For this set of features, we selected version 4 for use in the FFT implementation.

B. Synthesis Results for the FFT Architectures

The fully-sequential FFT architecture considered to achieve a sampling rate of 16 kilo-samples per second. Besides, the synthesis took into account 32, 64, 128, and 256-point FFT. The size of 16 bits was used for the input data and for the internal constants (twiddles). We adopted four strategies for these FFT architectures: (i) **Baseline** that is a multiplier-based butterfly implementation, (ii) **CORDIC**: that employs our best CORDIC optimization (v.4) that includes the butterfly embedding its twiddle factors calculation removing these coefficients storage, (iii) the multiplier-based butterfly employing the clock-gating technique, and finally (iv) combining both the CORDIC v.4 with clock-gating technique.

For a single processing unit (butterfly), and considering the sampling rate of f_s, the architecture operational clock frequency (Op. Freq.) can be calculated directly as being the interval between one sample and another divided by the total number of FFT operations (minus one-stage, i.e., the first, N = 2). The entire operation (complex rotations) FFT is the number of stages times the number of operations per stage. Therefore, according to (3), one can determine the operational clock frequency for sequential FFT architecture.

$$Op.Freq. = \frac{1/f_s}{(log_2(N) * N/2) - N/2} \quad (3)$$

According to (3), and the synthesis parameters defined above, we can observe, in Table III, the minimum operating clock frequency to achieve real-time. All the processing is in the interval between one sample and another (1 = F_s). Thus, this is the latency of the FFT operation sequentially.

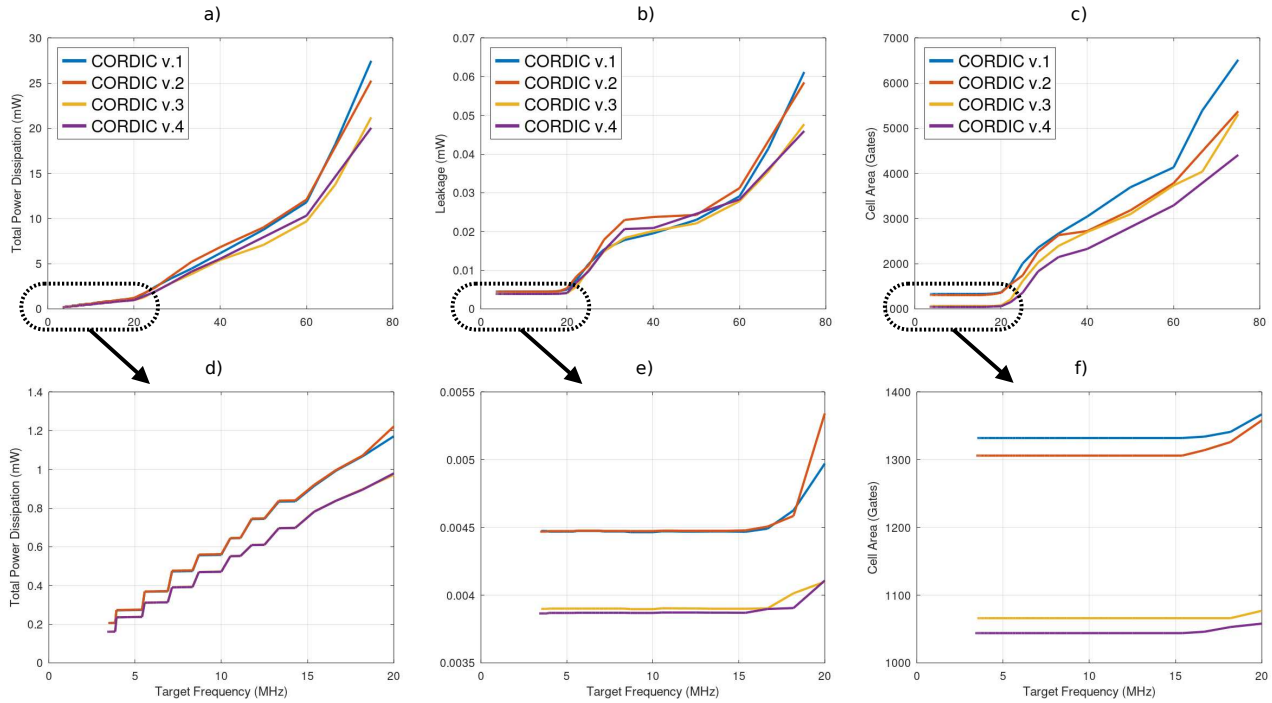

Fig. 9: Results for the hardwired CORDIC optimizations for synthesis at different clock frequency targets. a) and d) Power Dissipation. b) and e) Leakage. c) and f) Cell Area (Gates).

As can be seen, the power increases according to the number of points of the FFT. Also, with increasing frequency, the power increases. It occurs because the main power dissipation component (dynamic) is directly proportional to the operating frequency. The employing of our best CORDIC optimization (v.4) enables a power dissipation saving of 34% on average for N-point FFT (see Table III). The savings employing the clock-gating are 64% on average for N-point FFT. Notably, the clock-gating disables the clock in power-hungry blocks of the FFT architecture, which avoids unnecessary transitions. An important aspect to be observed in the synthesis of a circuit that uses the clock-gating technique is the static power consumption. It is possible to observe that clock-gating guarantees a great saving in power consumption when compared to its non-use, while the fraction dissipated by leakage current increases, although this fraction tends to decrease as the circuit size gets bigger. Finally, the results show that when combining our optimized CORDIC v.4 with the clock-gating technique, the power savings rises to 78.5% on average for N-point FFT. So, the most efficient implementation is the one that uses the combination of CORDIC and clock-gating, as can be seen in Table III.

Table III also presents the circuit area results for each proposal. The designs for each N-point are synthesized in their target clock frequency resulting in different circuit areas. We can observe that for 32-point FFT (i.e., relaxed target clock frequency), the baseline results in the smallest area. Higher target frequencies require high-performance multipliers in the butterfly version (i.e., the baseline). Therefore, starting from 64-point FFT (i.e., higher target clock frequency), the

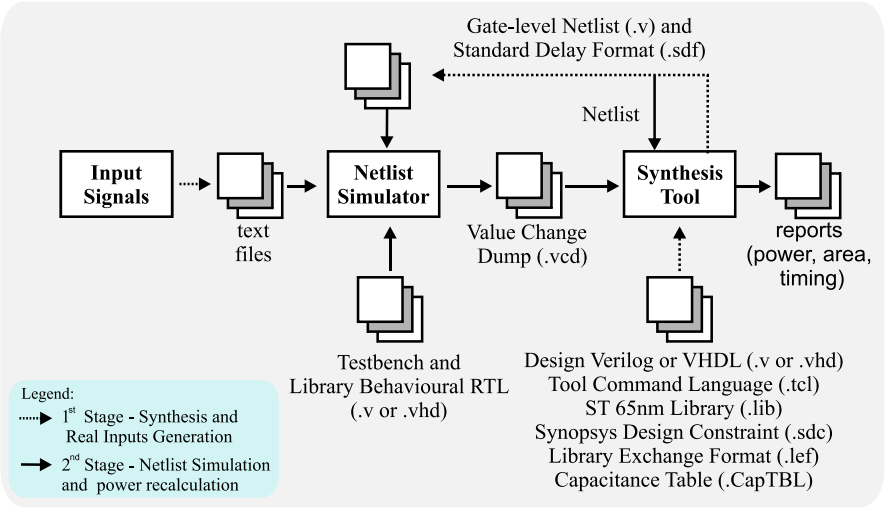

Fig. 10: Methodology for extracting power dissipation results.

Table III.: Circuit area and power dissipation (in mW) of the fully-sequential FFT with throughput for 16 ksps.

Points	Op. Freq.	Architecture	Area (Gates)	Leakage (mW)	Total Power (mW)	Savings (%)
32	1.024 MHz	**Baseline**	6143	0.04 (2.84%)	1.41	Reference
		CORDIC	7176	0.03 (3.96%)	0.96	31.91%
		Clock-gating (CG)	6113	0.04 (9.05%)	0.42	70.21%
		CORDIC + CG	7215	0.03 (11.11%)	0.27	80.85%
64	2.560 MHz	**Baseline**	10869	0.09 (2.57%)	3.31	Reference
		CORDIC	10098	0.06 (2.88%)	2.05	38.07%
		Clock-gating (CG)	10737	0.08 (8.32%)	0.95	71.30%
		CORDIC + CG	10148	0.06 (9.32%)	0.59	82.18%
128	6.144 MHz	**Baseline**	21424	0.18 (2.21%)	8.3	Reference
		CORDIC	16279	0.12 (1.97%)	5.83	29.76%
		Clock-gating (CG)	21016	0.17 (8.03%)	2.13	74.34%
		CORDIC + CG	16325	0.11 (8.50%)	1.27	84.70%
256	14.336 MHz	**Baseline**	44814	0.40 (1.23%)	32.46	Reference
		CORDIC	29722	0.24 (1.19%)	20.24	37.65%
		Clock-gating (CG)	43693	0.37 (1.93%)	19.32	40.48%
		CORDIC + CG	29774	0.22 (2.04%)	10.99	66.14%

Baseline represents the FFT version employing the butterfly based on multipliers (see Fig. 2).

CORDIC represents the FFT version employing the CORDIC v.4

Clock-gating represents the FFT version employing clock-gating in the shift-registers.

Area represents the cell area in 2-input NAND equivalent gates.

CORDIC-based version presents less circuit area than the baseline. The area savings increases to 24.0% in the 128-point and up to 33.6% in 256-point FFT. Clock-gating versions present slightly less circuit area than without clock-gating. The slight reduction is due to the replacement of individual enable logic of the registers by a grouped control of clock-gating logic as also reported in other works as [32,33].

C. Discussions and Comparisons with Related Work

Some works from the literature explored the CORDIC for the generation of the twiddle factors. Some of them use a ROM memory to store the angles, as in [27], that uses an address generation unit to generate the address of two input data and a rotation angle in ROM, required in the butterfly calculation.

The work in [30] proposes a CORDIC-based FFT architecture. The idea is the use of a new addressing scheme, where the twiddle factor angles follow a standard, increasing order, which can be generated by a simple accumulator. As the opposite, our best CORDIC core eliminates the angle generation. The aspect of no need to store angle values in ROM was explored in [26], as we do in our work. However, there no power results reported in [26] to prove the efficiency of it. The work in [28] calculates the angle values using the CORDIC cell for the required iteration. The angle values are fed to the proposed Rotator through the binary factor for binary information. However, there are also no power results to support the advantages of the proposed architecture.

In our work, it is no need to store the angle values. Also, there is the elimination of redundancies along with the iterations, which is significantly advantageous for power reduction. We prove the impact on power reduction of the proposed strategies by verifying the best CORDIC core in terms of both area and power dissipation. We used this best CORDIC core in a fully-sequential FFT based on shift-registers, which enabled us to use the clock-gating technique additionally.

The work in [8] also proposes the use of clock-gating and CORDIC. The work uses a radix 2-4-4-8-8 pipeline structure to achieve the low complexity of hardware and high reconfigurable flexibility. However, it used the clock-gating to reduce the power consumption of reading ROM, where the angles are previously stored. In our work, there is no ROM memory used, since one eliminates the values of the angles, and the clock-gating strategy is used in the registers of shift-register FFT implementation rather than in ROM memory.

The work in [20] proposes a ROM-free twiddle factor generator. According to the authors, by using the twiddle factor generator, the chip area and power consumption can be reduced significantly at the cost of an additional logic circuit. Our work tends to be more power-efficient because we use two strategies, the elimination of the angle accumulator processing and also the elimination of redundancies between iterations. Moreover, we also use the clock-gating approach in the FFT, which was not explored in [20].

It is worthwhile to mention that it is difficult to realize a direct comparison with the said work from the literature, based on ASIC, and that explore the use of CORDIC since they do not use the same technology as we are using in this work. Moreover, the work in [20] uses a split-butterfly, and the work in [8] uses a radix 2-4-4-8-8, while we are using a radix-2 butterfly for the CORDIC exploration.

VI. Conclusion

This paper presented dedicated architectures of the CORDIC algorithm and its application in specialized FFT architectures. We implemented unfolded architectures for the CORDIC algorithm in the pipeline and non-pipeline versions. Regarded to FFT, it was a fully-sequential configurable architecture, using CORDIC for the generation of twiddle factors. From the four architectures explored for the CORDIC, the one that presented the best results was it used the strategies of eliminating the accumulator angle and also eliminated redundancies between iterations. The fully-sequential FFT implementations used this best CORDIC configuration. The results showed that the use of CORDIC reduces the power substantially in the shift-register implementation of the FFT. However, the combination of using the CORDIC and clock-gating technique is the best option for a low-power FFT design.

VII. Acknowledgements

The authors would like to thank CNPq, CAPES and Fapergs Brazilian agencies for financial support to our research.

References

[1] G. Moore, "Moore's law," *Electronic Magazine*, vol. 38, no. 8, p. 114, 1965.

[2] G. Kumar, S. Sahoo, and P. Meher, "50 Years of FFT Algorithms and Applications," *Circuits Systems and Signal Processing*, vol. 38, pp. 5665–5698, 2019.

[3] F. Qureshi, *Optimization of rotations in FFTs - Thesis (PhD).* Linköping University Electronic Press, 2012.

[4] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[5] J. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on electronic computers*, no. 3, pp. 330–334, 1959.

[6] S. Dirliki, *A comparison of FFT processor designs (Master).* University of Twente, 2013.

[7] M. Kavitha, S. Ranjitha, and H. Suresh, "Review Paper on Efficient VLSI and Fast Fourier Transform Architectures," *International Journal of Engineering Sciences Research technology*, vol. 6, pp. 15–20, 2017.

[8] G. Liu and Q. Feng, "ASIC design of low-power reconfigurable FFT processor," in *7th International inproceedings on ASIC (ASICON2007)*, 2007, pp. 44–47.

[9] Y. Zhao, A. Erdogan, and T. Arslan, "A low-power and domain-specific reconfigurable FFT fabric for system-on-chip applications," in *19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[10] T. Ayhan, W. Dehaene, and M. Verhelst, "A 1282048/1536 point FFT hardware implementation with output pruning," in *22nd European Signal Processing inproceedings (EUSIPCO)*, Sep 2014, pp. 1–5.

[11] M. Garrido, S.-J. Huang, and S.-G. Chen, "Feedforward FFT Hardware Architectures based on Rotator Allocation," *IEEE Transactions on Circuits and Systems I: Regular Papers*.

[12] A. Tiwari and S. Pandey, "FPGA Implementation of FFT Processor in Xilinx," *International Journal of Engineering and Management Research*, vol. 6, pp. 134–138, 2016.

[13] A. Anbarasan and K. Shankar, "Design and implementation of low power FFT/IFFT processor for wireless communication," in *International inproceedings on Pattern Recognition, Informatics and Medical Engineering (PRIME)*, 2012, pp. 152–155.

[14] S. Noor, E. John, and M. Panday, "Design and Implementation of an Ultra low-Energy FFT ASIC for Processing ECG in Cardiac Pacemakers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, Apr 2019.

[15] A. Sridharan and A. Viji, "Low power hardware implementation of high speed FFT core," in *International inproceedings on Computer Engineering*, 2010, pp. 223–227.

[16] M. Garrido, S. Huang, S. Chen, and O. Gustafsson, "The Serial Commutator FFT," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, pp. 974–978, 2016.

[17] S. Sathasivam and G. Reddy, "ASIC implementation of high throughput FFT processor for scientific applications," *Indian Journal of Science and Technology*, vol. 9, no. 5, pp. 1–5, 2016.

[18] R. Teymourzadeh, *High Resolution Single-Chip Radix II FFT Processor for High-Tech Application*. 5772/66745. hal-01802070, 2017.

[19] J. Bhatt, "A Survey on CORDIC Based FFT Implementation on FPGA," *International Journal for Scientific Research Development*, vol. 1, pp. 2643–2645, 2014.

[20] T.-Y. Sung, H.-C. Hsin, and Y.-P. Cheng, "Low-power and high-speed CORDIC-based split-radix FFT processor for OFDM systems," *Digital Signal Processing*, vol. 20, pp. 511–527, 2010.

[21] M. El-Motaz, O. Nasr, and K. Osama, "A CORDIC-friendly FFT architecture," in *Wireless Communications and Mobile Computing inproceedings, IWCMC*, Aug 2014.

[22] A. El-Shafiey, M. Farag, M. El-Motaz, O. Nasr, and H. Fahmy, "Two-stage optimization of CORDIC-friendly FFT," in *IEEE International inproceedings on Electronics, Circuits, and Systems (ICECS)*, Dec 2015, pp. 408–411.

[23] K.Nagapramodh and K. S. Kumar, "Design and Implementation of Modified CORDIC based FFT using Vedic Multiplication Techniques," *International Research Journal of Engineering and Technology*, vol. 3, pp. 867–872, 2016.

[24] C. Korde, P. Malathi, S. Shelke, and M. Sharma, "Design of FPGA Based Radix-4 FFT Processor using CORDIC," *Journal of Innovation in Electronics and Communication Engineering*, vol. 5, pp. 56–62, Jun 2015.

[25] J. Shalini and R. Manjunatha, "FPGA based efficient n-point FFT architecture using CORDIC for advanced OFDM," *International Journal of Engineering Research and Technology*, vol. 11, no. 1, pp. 11–27, Jun 2018.

[26] J. Chikhaliya, C. Dave, and J. Tiwari, "Design and implementation of FFT processor using CORDIC algorithm," *International Journal of Innovative and Emerging Research in Engineering*, vol. 3, no. 4, pp. 143–148, 2016.

[27] A. Tang, L. Yu, F. Han, and Z. Zhang, "CORDIC-based FFT Real-time Processing Design and FPGA Implementation," in *12th International Colloquium on Signal Processing its Applications (CSPA2016)*, Mar 2016, pp. 233–236.

[28] M. Princy and A. Kumar, "Implementation of binary FFT using CORDIC algorithm," *Journal of Network Communications and Emerging Technologies (JNCET)*, vol. 7, no. 3, pp. 37–40, 2017.

[29] A. Fatima, M. Moinoddin, and A. Padma, "FPGA Hardware Implementation of a CORDIC-Based Radix-4 FFT Processor," *Indian Journal of Scientific Research(IJSR)*, vol. 17, pp. 291–295, 2018.

[30] K. Babu and D. Madhavi, "Design of Radix-2 64-Point FFT Processor Using CORDIC Algorithm," *International Journal for Scientific Research Development*, vol. 3, pp. 1898–1902, 2015.

[31] A. Sapper, L. Soares, E. Costa, and S. Bampi, "Exploring the combination of number of bits and number of iterations for a power-efficient fixed-point CORDIC implementation," in *IEEE International inproceedings on Electronics, Circuits, and Systems (ICECS)*, Dec 2017, pp. 302–305.

[32] Petracca, Michele and Carloni, Luca, "The Benefits of Using Clock Gating in the Design of Networks-on-Chip," 2011.

[33] Seidel, Ismael and Monteiro, Marcio and Güntzel, José Luís and Agostini, Luciano, "Squarer exploration for energy-efficient sum of squared differences," in *IEEE 7th Latin American Symposium on Circuits and Systems (LASCAS)*, 2016, pp. 327–330.