CASE STUDY

Low-power fast Fourier transform hardware architecture combining a split-radix butterfly and efficient adder compressors

Guilherme Ferreira1 | Guilherme Paim1 | Leandro M. G. Rocha1
Gustavo M. Santana1 | Renato H. Neuenfeld2 | Eduardo A. C. Costa2 | Sergio Bampi1

1Graduate Program in Microelectronics (PGMICRO), Federal University of Rio Grande do Sul (UFRGS), Brazil
2Graduate Program in Electronic Engineering and Computing, Catholic University of Pelotas (UCPel), Brazil

Correspondence
Guilherme Paim, Graduate Program in Microelectronics (PGMICRO), Federal University of Rio Grande do Sul, Brazil.
Email: gppaim@ieee.org

Abstract
Fast Fourier transform (FFT) is the most common low-complexity implementation of the discrete Fourier transform, intensively employed to process real-world signals in smart sensors for the internet of things. Butterflies play a central role as the FFT computing core data path since it calculates complex terms employing several multipliers. A low-power FFT hardware architecture combining split-radix decimation-in-time butterfly and 5-2 adder compressors (ACs) is proposed and implemented. The circuits are described in Verilog hardware description language and synthesized using the Cadence Genus synthesis tool. The circuits are mapped onto a 65-nm CMOS ST standard cell library. Results reveal that the proposed FFT hardware architecture using the split-radix butterfly is 13.28% more power efficient than the radix-4 one. The results further show that, by combining 5-2 AC within the split-radix butterfly, our proposal saves up to 43.1% of the total power dissipation considering the whole FFT hardware architecture, compared with the state-of-the-art radix-4 butterfly employing the adder automatically selected by the logic synthesis tool.

1 INTRODUCTION

In the current internet of things (IoT) era, low-power implementations of smart sensors are among the primary design concerns. Smart sensors are capable of locally processing the captured signal to reduce the burden on the RF (radio frequency) communication channel for transferring massive amounts of data. Signal processing in the sensor can drastically minimize the amount of energy consumed in both industrial [1] and consumer electronic wireless communication. For such applications, the ASIC design of digital signal processing (DSP) algorithms is of paramount importance to meet the energy consumption constraints.

Nowadays, power dissipation in VLSI DSP circuits has gained special attention, mainly due to the proliferation of battery-powered high-performance devices such as smartphones, portable monitors, and notebooks. Efficient DSP algorithms are applied to process data in smart IoT devices [2]. Among those, the fast Fourier transform (FFT) is one of the most widely employed to transform the signal from the time to the frequency domain. Smart sensors for IoT commonly employ FFT to perform the spectral analysis of the signals, as shown in [3]. When the energy-efficiency of devices with power restrictions becomes an issue, low-power VLSI architectures must be developed [4]. Therefore, designing low-power FFT hardware architectures is essential to maximize the efficiency of this type of smart sensors, as we are proposing here.

FFT is the most common low-complexity implementation of the DFT (discrete Fourier transform). FFT is often employed to support the data classification or compression in the IoT smart sensors [1]. The butterfly is the mathematical core of the FFT to perform complex operations. Butterflies perform the complex multiplication of input data by the twiddle factors [5] (i.e. appropriate coefficients). The twiddle factors are multiple values of \( N = 2 \), where \( N \) represents the number of FFT points. One of the main challenges in the FFT hardware architecture is the optimization of the butterfly by using efficient arithmetic operators, which is the goal of the architecture proposed here.

The FFT algorithm splits the processing into butterflies, performing the current stage by processing data from the previous ones. A single stage, split into butterflies, must
perform a significant part of the butterflies without using the last points of the input array [6]. The literature proposes different butterfly designs attending both of the characteristics mentioned, such as radix-2, radix-4, mixed-radix, and split-radix [7-13]. However, the split-radix algorithm requires the fewest operations among the other radix-2 and radix-4 algorithms [14]. The key idea of a split-radix butterfly is to mix decomposition of both radix-2 and radix-4 butterflies.

The hardware architecture of the split-radix butterfly enables the simultaneous addition of complex terms. For this task, employing adder compressors (ACs) is a promising alternative for low-power designs. ACs parallelization schemes allow the reduction of the hardware architecture’s internal switching activity, which proportionally reduces the dynamic power dissipation, compared with the combination of the conventional carry propagating adders [15,16].

By performing a careful analysis of the complex equations, composed of real and imaginary multiplications, additions, and subtractions, we propose a 5-2 AC in the butterfly of a split-radix with decimation in time (DIT). According to [14], the DIT FFT butterfly involves less computation time than the DIF A Q4 one. Therefore, we combine the optimized DIT split-radix butterfly within ACs in a fully-sequential 16-point FFT implementation to investigate the entire FFT architecture’s power efficiency. Furthermore, by implementing the DIT butterfly, we can compare our results with the literature solutions.

The hardware architectures were described in Verilog hardware description language (HDL) and synthesized for ASIC employing the standard cells of the ST 65nm CMOS process. We used a power extraction methodology flow to perform a realistic power estimation for considering the glitching. The results show that our split-radix butterfly proposal employing 5-2 ACs is more area and power efficient than both a baseline split-radix butterfly (without AC) and radix-4 butterflies. Furthermore, the FFT hardware architecture with the proposed split-radix butterfly results in higher area and power efficiency than the FFT employing the other aforementioned butterflies.

The main contributions of this work are as follows:

1. A hardware architecture for FFT employing a split-radix butterfly, which is more power-efficient than previous works.
2. Demonstration of the efficient design of 5-2 ACs within the split-radix butterfly used in the complete FFT hardware architecture.

This article is organized as follows: Section 2 presents a background on the FFT as well as an overview of split-radix butterfly and ACs. The most relevant works from the literature are also discussed in this section and compared with our proposals. Section 3 presents the architecture of the proposed FFT, combining the split-radix butterfly and the ACs. Section 4 shows the main results achieved by the proposed solutions. Finally, Section 5 highlights our conclusions.

2 | BACKGROUND

This section presents a background on the FFT, the split-radix butterfly structure, an overview of 5-2 AC, and finally, the most relevant works from the literature.

2.1 | FFT

FFT is one of the most important algorithms in signal processing and data analysis. The FFT can be applied to solve a myriad of problems such as speech and image processing, signal analysis, and communication systems. The proposed work in [14] shows a survey with the advances on the FFT algorithms and applications during the last 50 years.

The DFT is one of the most used mathematical applications in DSP and communication systems. The DFT is an accurate technique that converts temporal or spatial data into frequency domain data. The discrete version of the FT views both the time domain and frequency domain as periodic. In other words, it establishes a relationship between both the time and the frequency domain representation. However, the DFT is not computed directly due to both a large number of arithmetic operations and data transfers involved in its process. Thus, the FFT is preferred since it resorts to algorithms with lower computational cost, evaluating the DFT approximately and efficiently [14]. The FFT reduces the number of computations needed for a problem of size \( N \) from \( O(N^2) \) to \( O(N \log N) \).

One of the drawbacks of conventional FFT hardware architectures is the presence of multiplier blocks, which cost higher hardware complexity, increase the power consumption, and limit the maximum operating clock frequency as well. However, some techniques have solved this problem by using strategies such as the coordinate rotation digital computer [12] or even using more efficient ACs, such as in this work.

The most commonly used FFT algorithm is the Cooley-Tukey algorithm [5]. It is a divide-and-conquer algorithm for the computation of complex Fourier series. The main characteristic of this algorithm is to break the overall DFT into smaller DFTs. The basic idea uses the radix-2 butterfly block, that is, one calculates the FFT \( X(k) \) of a signal \( x(n) \) using (1), where \( W_N^k \) is the \( N \)-th twiddle factor, \( j \) is the imaginary unit, and \( N \) is the number of points of the FFT [5].

\[
X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn} \\
W_N = e^{-j\frac{2\pi}{N}} \tag{1}
\]

By referring to the FFT, one is not considering just a single algorithm or mathematical expression but a family of algorithms and methods to efficiently calculate the DFT. However, in all of these methods, the reasoning is similar and can be explained mathematically in several ways. Recent advances in the FFT algorithm take into account higher radix and split-radix.
Standard radix-2 algorithms use two half-length DFTs, and the radix-4 algorithms are based on four quarter-length DFTs. The split-radix algorithm uses both one half-length DFT and two quarter-length DFTs. It is possible because, in the radix-2 computations, the even-indexed points are independent of the odd ones. The split-radix algorithm uses the radix-4 algorithm to compute the odd-numbered points. Therefore, the \( N \)-point DFT is decomposed into one \( N/2 \)-point DFT and two \( N/4 \)-point DFTs [12]. The advantage of split-radix is that it has reduced a number of arithmetic computations compared with those of radix-4 and radix-2 FFT. The split-radix also has advantages such as a regular structure and no reordering of internal signals except for outputs.

2.2 | Split-radix with DIT overview

The split-radix butterfly’s main idea is twofold: to use one radix for one decimation product of a sequence and other radices for other decimation products of the sequence [12]. In practice, the split-radix FFT mixes radix-2 and radix-4 decomposition (as in Figure 1).

The DIT split-radix butterfly comprises two complex multiplications and other complex additions and subtractions (see Figure 1). The development of the complex equations (Figure 1) and the terms group into real and imaginary parts lead to the expressions (2)–(9). The split-radix implementation, with eight multipliers and 16 adders/subtractors, is derived directly from the previous expression’s factorization, shown in Figure 2.

\[
\begin{align*}
X_{0r} &= P_{0r} + W_{1r} \cdot P_{1r} - W_{1i} \cdot P_{1i} + W_{3r} \cdot P_{3r} - W_{3i} \cdot P_{3i} \\
X_{0i} &= P_{0i} + W_{1r} \cdot P_{1i} + W_{1i} \cdot P_{1r} + W_{3r} \cdot P_{3i} + W_{3i} \cdot P_{3r} \\
X_{1r} &= P_{2r} + W_{1r} \cdot P_{1i} + W_{1i} \cdot P_{1r} - W_{3r} \cdot P_{3i} - W_{3i} \cdot P_{3r} \\
X_{1i} &= P_{2i} - W_{1r} \cdot P_{1r} + W_{1i} \cdot P_{1i} + W_{3r} \cdot P_{3r} + W_{3i} \cdot P_{3i} \\
X_{2r} &= P_{0r} - W_{1r} \cdot P_{1r} + W_{1i} \cdot P_{1i} - W_{3r} \cdot P_{3r} + W_{3i} \cdot P_{3i} \\
X_{2i} &= P_{0i} - W_{1r} \cdot P_{1i} - W_{1i} \cdot P_{1r} + W_{3r} \cdot P_{3i} - W_{3i} \cdot P_{3r} \\
X_{3r} &= P_{2r} - W_{1r} \cdot P_{1r} - W_{1i} \cdot P_{1i} + W_{3r} \cdot P_{3r} + W_{3i} \cdot P_{3i} \\
X_{3i} &= P_{2i} + W_{1r} \cdot P_{1i} - W_{1i} \cdot P_{1r} - W_{3r} \cdot P_{3i} + W_{3i} \cdot P_{3r}
\end{align*}
\]

(2)

According to the literature [14], the split-radix yields an algorithm with about one-third fewer multipliers than those of the radix-2 FFT. The split-radix FFT has lower complexity than the radix-4, or any higher radix power-of-two FFT. We intend to reduce power in the split-radix butterfly by using efficient ACs.

2.3 | Overview of ACs

The AC structures are widely used in fast- and low-power multiplier architectures and are composed of exclusive-or (XOR) gates and multiplexers (MUX).

We propose to apply 5-2 AC [12] in the butterfly. Figure 3(a) shows the internal structure of the AC. Note that the 5-2 AC can be power advantageous because of its reduced critical path, composed of just four XOR gates for the simultaneous addition of five operands.

The 5-2 AC has seven inputs and four outputs, as presented in Figure 3(a). The seven inputs are the five operands \( A_1, B, C, D, E \), and two input carry signals \( C_{in0} \) and \( C_{in1} \). The four output signals are \( \text{Sum}, \text{Carry}, C_{out1}, \) and \( C_{out1} \), where the Carry and both \( C_{out0} \) and \( C_{out1} \) terms have 1-bit higher order than the Sum. We implemented the 5-2 compressor cell, as shown in Figure 3(a), at the logic level using six XOR and three MUX gates.

The implementation of an \( N \)-bit 5-2 AC, shown in Figure 3(b), uses \( N \) 5-2 compressor cells of Figure 3(a). Recombination of the partial \( \text{Sum}, \text{Carry}, C_{out1}, \) and \( C_{out2} \) terms must be realized along with the cells. An extra recombination line recombines \( \text{Sum} \) and Carry terms. This line is shown as a cascade of full adders in the Figure 3(b) as a ripple carry adder. In the VLSI synthesis results of this work the final sum’s recombination line is implemented by the (+) macro function operator selected by the synthesis tool.

2.4 | Related work

The low-power implementations of FFT algorithms from the literature try to optimize the entire FFT architecture by using pipelining, reusing butterflies, proposing sequential and semi-
parallel structures, or even reordering the twiddle factors, such as in [12] and [12]. Most of the prior works are based on radix-2 or radix-4 butterflies, such as in [9-13]. Other hybrid architectures use radix-2 and radix-4 butterflies, such as the split-radix butterfly, simultaneously.

Table 1 summarizes the related work compared to our work that explores split-radix butterfly employing ACs for the FFT VLSI design. The primary goal of the different solutions is to reduce the critical path and, consequently, improve the entire FFT architecture’s performance, as seen in [14], whose split-radix structure uses the radix-4 butterfly, but in which no power results are presented. The work in [27] also proposes a split-radix FFT based on the radix-4 butterfly. However, the results are only FPGA based and with no evidence of its efficiency since it was only compared against a 1024-point radix-2 FFT. A split-radix FFT based on a radix-2 butterfly was proposed in [28]. The modified radix-2 butterfly unit exploits the multiplier-gating technique to save the dynamic power at the expense of using more hardware resources. However, there are only results for the individual components of a 1024-point FFT in ASIC.

As the main characteristic of the split-radix butterfly is to reduce the number of arithmetic operators, the works in [29] and [30] propose a multiplierless architecture of the split-radix...
FFT algorithm using new distributed arithmetic. The work in [29] presents power results in both FPGA and ASIC implementations. However, its authors present no explanation about the methodology used for power estimation nor about the type of arithmetic operators used. On the other hand, the work in [30] uses parallel prefix adders in the split-butterfly, but its results are only for FPGA implementation and no power dissipation data is presented. The work in [31] also explores the use of distributed arithmetic (DA) for a 256-point split-radix FFT. The approach in [31] incorporates a method to overcome the result overflow problem introduced by the DA method.

The DA strategy has the advantage of substituting the multiplications in the butterfly. However, its bit-serial characteristic can negatively affect the performance operation since the total number of cycles required to complete the multiplication is proportional to the number of input bits [31]. Therefore, other strategies have tried to turn the split-radix FFT more efficient, such as in [32], proposing an ASIC-based shared-memory architecture. The work in [33] presents a shared-memory low-power split-radix FFT processor architecture, which is computed by using a modified radix-2 butterfly unit. Although both works claim that the solutions are low power, there are no comparison results to prove this, and only FPGA results are available. The work presented in [34] also presents a shared-memory-based split-radix FFT processor. Although [34] concludes that the proposed method reduces the dynamic power consumption at the expense of more hardware circuit area, there are no power dissipation figures in [34] to sustain it. The same for the work in [35], which proposes a low-power 128-point split-radix FFT, also lacking an investigation about the power dissipation.

The work in [36] promises the implementation of a low-power split-radix FFT processor. However, there is no evidence that the proposed approach is a low-power solution. Furthermore, there are only results in FPGA. An ASIC implementation of a 32-point split-radix FFT is presented in [37]. Although the authors claim that the proposed method has advantages in terms of chip area, short execution time, and power consumption, there are no low-power strategies applied nor comparisons to prove it.

A different scheme for split-radix FFT was presented in [38–41], where DIT radix-3 and radix-6 FFT can calculate a 12-point DFT. However, the results are only in FPGA with no power dissipation savings. A split-radix 4/8 FFT algorithm is proposed in [42], but with no power dissipation savings over the prior literature. Furthermore, the authors present no comments about the methodology for a realistic power extraction report.

Here we extend our prior work in [43], investigating the area and power efficiency of our new split-radix butterfly proposal in the whole FFT architecture. In contrast to the related work, we also show how the FFT architectures behave in a real-world scenario by submitting them to a thorough power-estimation methodology, as our Section 4.1 will address. It is noticeable in Table 1 that only our previous work in [43] used a detailed synthesis flow to obtain the results as we present here. Finally, none of the related work optimizes the FFT architectures employing the efficient ACs within the split-radix butterfly, as we are herein proposing. The work in [44] suggests an FFT using modified 4-2 and 7-2 ACs while omitting the entire FFT implementation. Moreover, no power dissipation investigation is made in that article to sustain the claimed efficiency of employing ACs. In summary, as the opposite of the literature solutions presented in Table 1, our work brings an FFT implementation with an optimized split-radix butterfly using efficient 5-2 ACs.

### 3.1 Split-radix butterfly employing ACs

The split-radix butterfly has a well-balanced structure, as shown in Figure 2. This aspect can contribute to glitching reduction, that is, reducing unnecessary switching activity, thus favorably impacting dynamic power reduction. Therefore, the split-radix structure is a prime candidate for a power-efficient implementation employing ACs.
By a careful analysis from expressions in (2), it is noticeable that the final values of $X_N$ terms are composed of the sum of five terms, making the 5-2 AC a suitable option for this implementation. The five terms to be used as inputs for the 5-2 AC are the first rows of multipliers that compute the $W_x \cdot P_y$ terms. Figure 5 presents the architecture proposed in [41] for the split-radix using 5-2 AC.

### 3.2 Employing the optimized split-radix butterfly within the FFT hardware architecture

Our proposed 16-point FFT implementation uses a split-radix butterfly, Figure 5-a, with DIT in its baseline structure. To draw a proper comparison between the FFTs, we also implemented an FFT using the radix-4 butterfly.

Implementing a fully parallel FFT for a large number of points is impractical. The current literature proposes several topologies to address this problem. Although there are several ways of implementing massive FFT architectures, most of the topologies use butterflies as their primary operator in a sequential structure.

As previously mentioned, the DIT algorithm is the basis for this architecture. First, to use this algorithm, we have to reorder the inputs to a bit-reversal indices order. Second, we separate the elements into groups. We split the first group according to the least significant bit. It is the same process for the second group, where we divide it according to the second least significant bit, and so on.

The developed architecture's data path is shared across split-radix and radix-4 butterfly types, differing only in the control signals (omitted in Figure 5 for simplification). The control path is composed of one finite state machine (FSM) for the split-radix and one for the radix-4 butterflies. The main difference is the number of clock cycles required to compute the FFT correctly in each butterfly design. The FSM is also responsible for enabling the correct MUX groups, register sets, and activating the ROM module's coefficients. As the twiddle factors depend only on the FFT size, we can find the real and imaginary parts of these twiddle factors earlier and store them in a ROM memory.

In the proposed sequential architecture, we use only one butterfly for processing the complex terms of the transform. A set of registers read the real (Figure 5-b) and imaginary (Figure 5-c) parts of the data. These same sets of controlled registers store the obtained results from the computation of each stage. The proposed architecture also uses four groups of MUX circuits targeting better control of the system. While the second line of the MUX (with A to D outputs) will feed the butterfly with data from the sets of registers at different periods, the last line of the MUX (with P1 to P3 outputs) group is responsible for organizing that data in the correct input ordering for the butterfly. Another group of these MUX circuits, the first line of MUX, receives the data from the TRUNC1Q13 and orders them before storing them in the register sets.

The TRUNC modules, connected to the butterfly outputs, are composed mainly of buffers. As it corresponds to a 16-bit architecture, these blocks truncate its inputs to the most significant 16 bits of the butterfly calculation results. These 16 bits represent the most significant part of the fractional value between -1 and 1. Finally, Figure 5-d shows the register with control, which allows greater control over the register group, enabling the use of previously calculated results or the new FFT inputs each 15 clock cycles. Figure 5-d shows the internal structure of the controlled-input set of registers.

### 4 RESULTS AND DISCUSSIONS

This section presents the results of our work. We show the synthesis results to both the investigated butterflies, and it is employing in FFTs. This section begins gives the technical details about the power extraction methodology purposefully used to obtain a realistic estimation. Our method is carefully described to assure the reproducibility of the results herein presented.

#### 4.1 Power extraction methodology flow

A realistic power estimation methodology is used here, which considers the signal transitions and the circuit's node
capacitance loads estimated by a pre-placement of the standard cells. Hence, our power estimation is layout aware, as it uses the information on gates and interconnections delays as inputs for the synthesis tool [45].

1. Standard delay format (SDF) generation occurs after the first initial synthesis. This format enables the precise estimation of temporal glitches since it considers the delay for all transitions. The SDF is highly recommended for a realistic power dissipation estimation. The results presented here show the difference between power results with and without considering the delay files.

2. Physically aware layout estimation (PLE) mode estimates the length of all nets and considers the load capacitance effects in the power dissipation, finding a relatively pessimistic layout estimation. The library exchange format (LEF) files for all cells are used in the estimate. This file contains the library’s physical layout information. The LEF macro consists of the internal logic cell capacitance, and the tech LEF includes the process metal capacitance for the interconnection capacitance estimation [46]. Another file with the same information as LEF is named CapTable and contains the routing capacitance table. The CapTable file is more precise than LEF, however, since it considers the process variations [46]. Finally, the standard cell library files (in liberty format) are employed for power dissipation estimation. These files contain the information of the power dissipation for each slew rate and load capacitance. The results given here are from industrial-strength cell libraries from ST 65 nm process [47].

To provide a more precise power estimate, we generate the stimulus format from the gate-level netlist simulation, running a testbench with real signal inputs. The simulation runs on the gate netlist, the Verilog generated after the first logic synthesis [47]. This method guarantees area and power precise values, since the simulation model is the same as the final circuit. The stimulus file formats supported by the commercial tool are the value change dump, the toggle count format, or the switching activity interchange format.

A new split-radix butterfly using 5-2 AC is herein proposed and it is compared with other solutions from the literature in the context of the full FFT VLSI design. The circuits were described in Verilog HDL (Hardware Description Language) and synthesized using the Cadence Genus synthesis tool [48]. The circuits were mapped onto a 65-nm CMOS ST standard cell library [49] using 1.0 V voltage supply and at the 125°C temperature corner. For fair power comparisons, a 10 MHz operating clock frequency was set for all butterfly circuits compared in Table 2, while considering 100k random input test vectors and real logic circuit delays. We present a comparison regarding gate count (relative to 2-input NAND gates), total cell area, critical path delay, and power dissipation initially for the butterflies only in Table 2.

### 4.2 Butterflies’ synthesis results comparison

The single butterflies were individually synthesized to investigate our proposal’s efficiency compared with others previously
The following butterflies were designed, synthesized, and evaluated at 65 nm, according to the method previously described:

1. Radix-4 butterfly in [10]
2. Radix-4 butterfly in [14]
3. Original split-radix butterfly in [34] (without ACs)
4. Our proposal: split-radix butterfly with 5-2 AC

Table 2 shows circuit area and critical path delay synthesis results for these four butterflies. The radix-4 architecture is composed of 12 multipliers and 22 adder/subtractors [8], which results in a total area of 44078.84 μm². On the other hand, the optimized version of the radix-4 decreases the number of multipliers and adders, enabling a reduction of 7.25% in its total area. As the split-radix butterfly has only 8 multipliers and 16 adders/subtractors, it results in a total area of 30657.64 μm², which reveals a reduction of 30.45%, compared with the original radix-4 butterfly. Note that the impressive decrease in the arithmetic operators has allowed the split-radix butterfly to also present an area reduction of around 25% compared with the optimized radix-4 butterfly.

The use of ACs has been advantageous for the split-radix butterfly, with an impressive reduction in the area, as can be seen in Table 2. The reduction that occurs because the ACs have a simplified structure based on XOR and MUX gates only, with a slight delay penalty, as shown in Figure 3. We could also substitute 16 adders/subtractors in the original split-radix butterfly by eight 5-2 ACs in the optimized one.

For a realistic power dissipation evaluation, we have included results considering the SDF files, as shown in Table 3. The SDF files are essential to enable a precise estimation of the temporal glitches, which is crucial for the dynamic power dissipation.

To the best of our knowledge, works in the current literature lack a thorough power analysis of the butterflies or even the entire FFT, considering a realistic power extraction. A meaningful estimation is necessary, at least based on vector-driven timed simulations capable of capturing logic glitches, or on realistic layout estimation, all of which affect the power dissipation.

To show the importance of presenting results with realistic power extraction, we offer in Table 3 power results with and without the SDF file. Note that, as the opposite of the original radix-4 and the split-radix architectures, the optimized radix-4 butterfly is an irregular structure, which makes it prone to higher glitching activity [8]. When synthesized without the SDF files, the optimized radix-4 saves 5.36% in power dissipation. However, when the synthesis with the SDF files is taken into account, its dynamic power increases more than four times, which indicates a massive influence of glitching activity in the entire architecture.

Although the proposed split-radix butterfly with 5-2 AC has a slightly longer critical path, it presents substantial power improvements by more than 30% reduction compared with the original split-radix. Compared with the original radix-4 butterfly, the proposed butterfly architecture shows an even more significant improvement in power, with savings of 55.97% and up to 47.82%, without and with SDF analysis, respectively. This is due to our AC optimized design, which uses a reduced number of transistors with transmission gate MUXes or pass-transistor logic [50].

### Table 2: Butterflies comparison: Circuit area and critical path delay synthesis results for 65 nm CMOS

| Butterfly circuits | Gate count (NAND2X1) | Circuit area (μm²) | Savings (%) | Delay (ns) | M. Freq.(MHz) |
|--------------------|---------------------|--------------------|-------------|------------|--------------|
| Radix-4 [10]       | 9930                | 44078.84           | -           | 18.88      | 52.96        |
| Radix-4 [14]       | 8783                | 40881.88           | 7.25%       | 23.13      | 43.23        |
| Split-radix [34]   | 6702                | 30657.64           | 30.45%      | 16.79      | 59.55        |
| Proposed Split-radix w/5-2 Compressors | 4960                | 21578.44           | 51.05%      | 17.92      | 55.79        |

### Table 3: Power dissipation comparisons @ 10 MHz

| Butterfly circuits | Power dissipation (μ W) | Leakage | Dynamic | Total | Savings | Leakage | Dynamic | Total | Savings | Glitching |
|--------------------|-------------------------|---------|---------|-------|---------|---------|---------|-------|---------|-----------|
|                    | Without SDF (imprecise analysis) [9] | 27.09   | 367.99  | 395.08| -       | 27.09   | 992.08  | 1019.17| -       | 61.23%    |
|                    | With SDF (precise analysis)            | 26.06   | 347.86  | 373.92| 5.36%   | 26.06   | 1541.37 | 1567.44| -53.80% | 76.14%    |
|                    |                                        | 19.33   | 290.68  | 310.01| 21.53%  | 19.33   | 766.34  | 785.67 | 22.91%  | 60.54%    |
|                    |                                        | 14.99   | 157.96  | 173.96| 55.97%  | 14.99   | 516.83  | 531.83 | 47.82%  | 67.29%    |

Results for the related work’s FFT architecture re-synthesized for the same process technology and running at same clock frequency of our work.
4.3 | Synthesis results for the FFT comparison

Results for the entire 16-point FFT architectures, with the butterflies designed in the previous section, are presented next. The architectures were synthesized for a maximum clock frequency of 150 MHz. As the butterfly results made clear the importance of using a realistic power extraction method, we present power results using a precise analysis with the SDF file.

The FFT architecture with the split-radix butterfly is the most efficient over the radix-4 butterfly in both circuit area and power dissipation. When considering the use of 5-2 AC, the results are further improved, proving it to be the most power-efficient combination among all structures. It occurs due to the optimal combination of the low-power aspect presented by the efficient AC and the reduced amount of arithmetic operators presented by the split-radix butterfly structure.

Reducing two multipliers leads to the smaller area presented by the optimized split-radix butterfly, compared with the original one, even though the optimized structure shows two more adders/subtractors. It has an impact on reducing the area of the FFT architecture, as can be compared in Table 4. It is noticeable that the area saving is smaller in the full FFT architecture with our optimized split-radix butterfly (Table 4) compared with the single butterfly (Table 2). This occurs because, in the FFT implementation, other blocks contribute to the power dissipation (registers, MUX, ROM, control state machine). Mainly, the ROM and the state machines differ in the FFT implementations with either split-radix or radix-4 butterflies.

As the implemented FFT is a fully sequential structure, with the processing of one butterfly per clock cycle, the use of an optimized butterfly has a considerable contribution to the power reduction. In this aspect, using a split-radix butterfly has been advantageous for power reduction, as shown in Table 4. This reduction is more significant when the FFT uses the optimized split-radix butterfly. As it has a longer critical path, the optimized butterfly presents a slightly higher delay than the original structure. Despite the critical delay disadvantage, the optimized split-radix butterfly offers gains in terms of dynamic power, which leads to the reduced total power presented by its design. The use of ACs with optimized internal structure with XOR and MUX only contributes to the power dissipation reduction.

4.4 | Discussions and comparisons with the literature

To compare our proposed FFT directly with others from the literature is not easy since, to the best of our knowledge, there are no works that present VLSI power results only for the split-radix butterfly. Then, here, we synthesized prior work architectures (as shown in the top 3 lines of Table 4) with the same 65 nm VLSI cell-library, and estimated area and power for four of them with the same method. Table 4 summarizes area and power results from prior published entire FFT architectures—not just the butterflies—to enable comparing them with our FFT proposed architecture.

According to the results in Table 4, it is noticeable that our FFT with split-radix butterfly is more area and power efficient than the solutions with radix-4 butterfly. Compared with the FFT with the split-radix butterfly from the literature [34], our design is more circuit area and power efficient for the same number of 16 points. It occurs because our split-radix butterfly uses efficient 5-2 ACs. Some other works have presented power results for the entire FFT with split-radix butterfly, but some are FPGA-based reports, such as in [27–29]. The work in

| FFT circuits | FFT Size | Gate Count (NAND2X1) | Circuit Area (mm²) | Technology (µm) | Frequency (MHz) | Power dissipation (mW) | With SDF (precise analysis) |
|--------------|----------|----------------------|-------------------|----------------|----------------|-----------------------|--------------------------|
|              |          |                      |                   |                |                | Leakage               | Dynamic                  | Total                    | Savings                |
| **Radix-4**  | 16       | 12,907               | 0.0681            | 0.065          | 150            | 0.067                | 27.94                    | 28.01                    | -                      |
| **Radix-4**  | 16       | 12,957               | 0.0739            | 0.065          | 150            | 0.077                | 31.69                    | 31.77                    | –13.42%                |
| **Radix-32** | 1024     | ×                    | 0.198             | ×              | 180            | ×                     | ×                        | 8.88                    | -                      |
| **Split-radix in [6]** | 2048 | ×                    | ×                 | ×              | ×              | ×                     | ×                        | 20.00                    | -                      |
| **Split-radix in [5]** | 1024 | 290,533              | ×                 | 0.045          | 100            | ×                     | ×                        | 102.19                   | -                      |
| **Split-radix in [5]** | 32   | ×                    | 1.063             | 0.18           | 100            | ×                     | 84.18                    | ×                        | 401                    | -                      |
| **Split-radix in [5]** | 256  | ×                    | 11                | 0.18           | 100            | ×                     | ×                        | ×                        | 402                    | -                      |
| **Split-radix in [5]** | 16   | 9175                 | 0.0442            | 0.065          | 150            | 0.059                | 24.23                    | 24.29                    | 13.28%                 |
| **Split-radix in [5]** | 32   | 979,228              | 32.358            | 0.18           | ×              | 0.044                | 242.73                   | 242.78                   | -                      |
| **Split-radix in [5]** | 64   | ×                    | 0.61              | 0.045          | 200            | ×                     | ×                        | 404.39                   | -                      |
| **Our proposal** | 16   | 9388                 | 0.0418            | 0.065          | 150            | 0.054                | 15.88                    | 15.94                    | 43.10%                 |

*Results for the related work’s FFT architecture re-synthesized for the same process technology and running at same clock frequency of our work.
[6] presents a reduced power value, but it only shows full results for the entire FFT in FPGA. It is known that for a realistic, precise, and low-power extraction method, FPGAAs are not the most appropriate technology.

Regarding ASIC-based FFT, there are solutions in the literature that present power reports, such as in [28], which proposes a 1024-point FFT with a split-radix butterfly based on a radix-2 one. In the scheme, clock-gating registers are put in the multiplier path to prevent unnecessary switching activity. A few registers are placed at the address port of memory banks to synchronize the whole design. Therefore, the method reduces the dynamic power consumption at the expense of more hardware resources. Here, the switching activity is reduced with no additional hardware resources since the ACs present small switching activity stemming from the reduced number of transistors in the XOR and MUX gates. Therefore, our FFT hardware architecture proposal presents a reduced power dissipation compared to the other ASIC solutions of the literature as well as a competitive maximum clock frequency of 150 MHz.

The literature presents different split-radix butterflies, such as in [42], which proposes a split-radix based on radix-4 and radix-8 butterflies. The work offers a tool that can generate the VHDLA Q19 descriptions of the required N-point fully parallel split-radix/8 FFT design that can be synthesized on an ASIC and/or FPGA platform. However, the developed architectures aim to achieve high-throughput and low-latency demands, without savings in power dissipation possible with other butterflies. There is a high power dissipation presented by this solution, synthesized for the Nangate 45 nm open-source process library and operating at 200 MHz.

Our work shows the methodology employed for a realistic power dissipation extraction for either butterflies alone or the entire FFT. This methodology allows for improving the reproducibility and comparability with prior research architectures from others. We use the more precise, placement-aware SDF file to include real delays, thus precisely considering the glitching contributions to the power results. We showed the power dissipation reduction when employing efficient ACs into the split-radix butterfly within the FFT.

5 | CONCLUSION

We presented a low-power FFT hardware architecture combining 5-2 AC within a split-radix butterfly. We applied a realistic methodology for the power dissipation estimation, including the circuit glitching activity. The results showed that the FFT employing the proposed split-radix butterfly architecture leads to considerably less power dissipation than original and optimized radix-4 butterflies. Moreover, combining the split-radix butterfly with 5-2 AC increased power savings even more in both the split-radix butterflies and the entire FFT architecture, reaching up to 47.28% and 43.10% savings, respectively. Therefore, our VLSI FFT architecture proposal contributes for maximizing the power efficiency of smart sensors for IoT that commonly employ FFT to perform the spectral analysis of the signals.

ACKNOWLEDGEMENTS
This study was financed in part by CNPq, CAPES, and Fapergs Brazilian agencies. We thank them for supporting our research.

ORCID
Guilherme Ferreira © https://orcid.org/0000-0001-5706-6068
Guilherme Paim © https://orcid.org/0000-0001-7809-9563

REFERENCES
1. Yasuura, H., et al.: Smart Sensors at the IoT Frontier. Springer International Publishing Cham, Switzerland (2017)
2. Buzachis, A., et al.: Development of a smart metering microservice based on fast Fourier transform (FFT) for edge/internet of things environments. In: IEEE 3rd International proceedings on fog and edge computing (ICFEC), pp. 1–6. IEEE (2019)
3. Raj, S.: An efficient IoT-based platform for remote real-time cardiac activity monitoring. IEEE Trans. Consumer Electron. 66(2), 106–114 (Dec 2020)
4. Henkel, J., et al.: Ultra-low power and dependability for IoT devices (Invited paper for IoT technologies). In: Design, automation & test in Europe conference & exhibition (DATE), pp. 954–959. IEEE (2017)
5. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19, 297–301 (1965)
6. Lo, P-C., Lee, Y-Y.: A real-time implementation of the split-radix FFT algorithm for efficiently constructing local butterfly modules. Signal Process. 1(1), 291–299 (1998)
7. Duhamel, P., Hollmann, H.: Split radix FFT algorithm. Electron. Lett. 20(1), 14–16 (1984)
8. Takahasi, D.: An extended split-radix FFT algorithm. IEEE Signal Process. Lett. 8(5), 145–147 (May 2001)
9. Takala, J., Punkka, K.: Butterfly unit supporting radix-4 and radix-2 FFT Proceedings of the 2005 international TICSP workshop on spectral methods and multirate signal processing, Riga, Latvia, SMSP, vol. 30, pp. 44–54 (2005)
10. Takala, J., Punkka, K.: Scalable FFT processors and pipelined butterfly units. J. VLSI Signal Process. Syst. Image Video Technol. 43(2), 113–123 (2006)
11. Neuenfeld, R., Fonseca, M., Costa, E.: Exploiting addition schemes for the improvement of optimized radix-2 and radix-4 fft butterflies. In: IEEE 8th Latin American symposium on circuits & systems, pp. 1–4 (2017)
12. Elango, K., Munianti, K.: Hardware implementation of FFT/IFFT algorithms incorporating efficient computational elements. J. Electr. Eng. Technol. 14, 1717–1721 (2019)
13. Elango, K., Munianti, K.: VLSI Implementation of an area and energy efficient FFT/IFFT core for MIMO-OFDM applications. Ann. Telecommun. 75(5–6), 215–227 (2020)
14. Lin, S-J, Chung, W-H.: The split-radix fast Fourier transforms with radix-4 butterfly units. In: Signal and information processing association annual summit and proceedings (APSIPA), pp. 1–5 (2013)
15. Diniz, C.M., et al.: Evaluating the use of adder compressors for power-efficient HEVC interpolation filter architecture. Analog Integrated Circuits. Signal Process. 89(1), 111–120 (Oct 2016)
16. Paim, G., et al.: Exploring high-order adder compressors for power reduction in sum of absolute differences architectures for real-time UHD video encoding. J. Real-Time Image Process. 1(1), 1–20 (2020)
17. Meher, P.K., et al.: Efficient VLSI architecture for decimation-in-time fast Fourier transform of real-valued data. IEEE Trans. Circuit. Syst. Regular Paper. 62(12), 2836–2845 (Dec 2015)
18. Kumar, G., Sahoo, S., Meher, P.: 50 years of FFT algorithms and applications. Circuit. Syst., Signal Process. 38, 5663–5698 (May 2019)
19. Sorensen, H., et al.: Real-valued fast Fourier transform algorithms. IEEE Trans. Acoust. Speech Signal Process. 35(6), 849–863 (1987)
20. Choudhary, P., Karmakar, A.: CORDIC based implementation of fast Fourier transform. In: 2nd International conference on computer and communication technology (ICCT), pp. 550–555 (2011)

21. Soni, M., Kunthe, R.: A General Comparison of FFT algorithms. 9th National Conference, Pioneer Journal for IT and Management (2011)

22. Chang, C.-H., Gu, J., Zhang, M.: Ultra low-voltage low-power CMOS 4-2 and 5-2 ACs for fast arithmetic circuits. IEEE Trans. Circuit. Syst. I Reg. Paper. 51(10), 1985–1997 (2004)

23. Sophy, P.A., et al.: Analysis and design of low power radix-4 FFT processor using pipelined architecture. In: International conference on computing and communications technologies (ICCCT), pp. 227–232 (2015)

24. da Luz, A.G., da Costa, E.A.C., Ghissoni, S.: Reducing the Hamming distance of encoded FFT twiddle factors using improved heuristics algorithms. In: IEEE 4th Latin American symposium on circuits and systems (LASCAS), pp. 1–4 (2013)

25. Neuenfeld, R., Fontseca, M., Costa, E.: Design of optimized radix-2 and radix-4 butterflies from FFT with decimation in time. In: IEEE 7th Latin American symposium on circuits & systems, pp. 1–4 (2016)

26. Garrido, M., Huang, S., Chen, S.: Feedforward FFT hardware architectures based on rotator allocation. IEEE Trans. Circuit. Syst. 1. 65(2), 581–592 (2018)

27. Patil, S., Sharma, M.: A 2048-point split-radix fast Fourier transform computed using radix-4 butterfly units. Intl. J. Recent Technol. Eng. (IJRTE). 8(3), 2043–2046 (2019)

28. Qian, Z., Margala, M.: Low-power split-radix FFT Processors using radix-2 butterfly units. IEEE Intl. Trans. Very Large Scale Integr. (VLSI) Syst. 24(3), 1–5 (2016)

29. Das, A., et al.: Efficient VLSI architectures of split-radix FFT using new distributed arithmetic. Intl. J. Soft Comput. Eng. (IJSCE). 3(1), 264–27 (2013)

30. Lagari, N., Amusudha, K.: VLSI implementation of efficient split radix FFT based on distributed arithmetic. In: International conference on green computing communication and electrical engineering (ICGCCEE), pp. 1–5 (2014)

31. Josh, S., Paily, R.: Distributed arithmetic based split-radix FFT. J. Signal Process. Syst. 75(1), 85–92 (2013)

32. Qian, Z., Nasiri, N., Segal, O., Margala, M.: FPGAs implementation of low-power split-radix FFT processors. In: 24th International proceedings on field programmable logic and applications (FPL), pp. 1–2 (2014)

33. Venu, P., et al.: Design and implementation of an area efficient split-radix FFT processors using radix-2 butterfly units. Intl. J. Adv. Res. Trend. Eng. Techno. (IJARTET). 4(7), 46–53 (2017)

34. Avinash, Managuli, M., Babu, D. S.: Low-power split-radix FFT processors. Intl. J. Adv. Sci. Eng. 6(6), 1253–1259 (2017)

35. Shilpha, G.: Low Power 128 point split radix FFT for LTE application. Intl. J. MC Square Sci. Res. 6(1), 148–157 (2014)

36. Ramamohan, Nancharaiah, V., Patnayakuni, V.: Implementation of low-power split-radix FFT processor. Intl. J. Manage., Technol. Eng. 8(12), 5818–5824 (2018)

37. Reddy, N., et al.: ASIC implementation of high speed fast Fourier transform based on split-radix algorithm. In: International conference on embedded systems, pp. 22–27 (2014)

38. Nancy, J., Bharath, S., Wilson, J.: Implementation of split radix algorithm for length 6-DFT using VLSI’. Intl. J. Innovat. Res. Technol. Sci. (IJIRTS). 2(3), 45–51 (2014)

39. Maladri, P., Jayanayudu, D., Mahendra, G.: Implementation of split radix algorithm for 12-point FFT & IFFT. Intl. J. Adv. Res. Electr. Electron. Instrument. Eng. 3(10), 12528–12534 (2014)

40. Naresh, G., Sruth, S.: Design and implementation of split radix algorithm for length-6 DFT using VLSI and FPGA. Intl. J. Res. Comput. Appl. Robot. 2(7), 36–45 (2014)

41. Harshita, B., Reddy, R.: Implementation of split radix algorithm for 12-point FFT. Intl. J. Mag. Eng. Technol. Manage. Res. 3(9), 1135–1140 (2016)

42. Raut, A., Pasha, M., Masud, S.: Towards design and automation of a scalable split-radix FFT processor for high throughput applications. Microprocess. Microsyst. 65, 148–157 (2019)

43. Santana, G.M., et al.: Using efficient adder compressors with a split-radix butterfly hardware architecture for low-power IoT smart sensors. In: International conference on electronics, circuits and systems (ICECS), pp. 486–489 (2017)

44. Daryani, M., Kumar, K.: Fast Fourier transform utilizing modified 4:2 & 7:2 compressor. Intl. J. Eng. Res. Dev. 11(5), 23–28 (2015)

45. Silveria, B., et al.: Power-efficient sum of absolute differences hardware architecture using adder compressors for integer motion estimation design. IEEE Trans. Circuit. Syst. 1 Regular Paper (99), pp. 1–12 (2017)

46. Manual: Genus Synthesis Flows. http://www.cadence.com.

47. White Paper: Power-Synthesis Option. http://www.cadence.com

48. Cadence EDA tools. http://www.cadence.com

49. ST LP 65nm standard cell Library. http://www.st.com

50. Zimmermann, R., Fichtner, W: Low-power logic styles: CMOS versus pass-transistor logic. IEEE J. Solid-State Circuit. 32(7), 1–12 (1997)

How to cite this article: Ferreira G, Paim G, Rocha LMG, et al. Low-power fast Fourier transform hardware architecture combining a split-radix butterfly and efficient adder compressors. IET Comput. Digit. Tech. 2021;15:230–240. https://doi.org/10.1049/ cdt.2.12015