EURASIP Journal on Advances in Signal Processing

RESEARCH **Open Access**

Optimized power and speed of Split-Radix, Radix-4 and Radix-2 FFT structures

Mahsa Shirzadian Gilan[1*] and Behrouz Maham[2]

This study was supported by the Ministry of Science and Higher Education of the Republic of Kazakhstan through project AP13068587, titled "57-65 GHz Wireless Communication Front-End for Secure 5G Applications".

*Correspondence: m.shirzadian@uok.ac.ir

[1] Faculty of Engineering, University of Kurdistan, Sanandaj, Iran
[2] School of Engineering, Nazarbayev University, Astana, Kazakhstan

Abstract

Fast Fourier transform (FFT) is a fundamental building block for digital signal processing applications where high processing speed is crucial. Resource utilization in implementing FFT structures can be minimized by optimizing the performance of multipliers and adders used within the design. FFTs are also widely used in various machine learning algorithms. To achieve increased processor efficiency and reduced resource utilization, we propose a hardware design for Radix-2, Radix-4, and Split-Radix FFT architectures that utilizes a novel parallel prefix adder. This design offers lower power consumption, smaller chip area, and faster operation compared to existing architectures. Our performance analysis focuses on metrics such as power consumption, clock speed, and hardware complexity for Radix-2, Radix-4, and Split-Radix FFT algorithms implemented with the proposed adder. We compare these metrics using our proposed arithmetic structure against existing adder designs. The results indicate that the Split-Radix FFT architecture achieves lower power consumption and smaller chip area compared to Radix-4 and Radix-2 methods. Additionally, the Split-Radix FFT exhibits a higher clock speed. Therefore, based on these findings, the Split-Radix algorithm appears to be a compelling choice for implementation on field-programmable gate arrays due to its high speed and lower power consumption.

Keywords: Power consumption, Speed, Split-Radix, Radix-4, Radix-2, Parallel prefix adder

1 Introduction

The Fourier transform is an essential method in most digital signal processing (DSP) applications, including image and signal processing, telecommunications, and optics. However, their widespread use comes at the cost of high computational demands. This is particularly true for the discrete Fourier transform (DFT), where the number of computations required grows to M^2 for a transformation size of M. As M increases, the calculations become exponentially complex [1–3]. To address this challenge, the fast Fourier transform (FFT) was developed as an established technique in signal processing. It significantly reduces the complexity of DFT calculations from M^2 to $M \log_2 M$, making it a more efficient approach for large datasets.

Based on existing literature, numerous FFT algorithms, such as the fast Hartley transform (FHT), have been proposed. However, Split-Radix, Radix-4, and Radix-2 algorithms remain the most widely employed [4, 5]. In DSP, the discrete Fourier transform (DFT) transforms a signal, denoted by $y(n)$ in the time domain, to its frequency domain representation, $Y(k)$. Both $y(n)$ and $Y(k)$ can be complex or real numbers. Calculating $Y(k)$ for each frequency value (k) requires M complex multiplications and M complex additions. Since there are M values of $Y(k)$, an M-point DFT necessitates M complex addition and multiplication blocks. Furthermore, the total number of addition and multiplication operations to compute this M-point DFT scales quadratically with M (M^2). This complexity becomes significant when dealing with large datasets. It is important to note that convolution in the time domain corresponds to multiplication in the frequency domain [5, 6]. Fourier transforms decompose signals into their fundamental sinusoidal components of various frequencies across the entire sequence. In contrast, wavelet transforms break down signals into localized components that have a defined location in both time and frequency. Some works have explored the combination of these techniques, known as fractal-wavelet analysis [7–12].

FFT algorithm leverages a key strategy called "divide-and-conquer." This method breaks down the computation of a large DFT into calculations of smaller DFTs. To understand its efficiency, let us consider a signal $y(n)$ with an even length, i.e., M is divisible by 2. Dividing $y(n)$ into two equal-sized sub-arrays requires performing some multiplication and addition operations on each sub-array to compute their respective DFTs [13, 14]. Moreover, calculating the DFTs of both sub-arrays separately would seem to require twice the number of operations for the original DFT. The power of the FFT algorithm lies in its ability to exploit clever mathematical relationships between the DFTs of the sub-arrays and the DFT of the original signal, $y(n)$. By utilizing these relationships, the FFT significantly reduces the total number of calculations needed compared to the conventional approach. This translates to faster computation times, making FFT a cornerstone for various DSP applications.

The classification process in FFT involves dividing the signal $y(n)$ into smaller sub-arrays and calculating $Y(k)$ from the DFTs of these sub-arrays [15]. This divide-and-conquer strategy has a dramatic reduction in calculations when the length of the sequences is a power of 2. In a Radix-2 FFT, the sequence $y(n)$ with length M is divided into two sub-arrays based on their even or odd indices [13, 16]. One sub-array consists of all even-indexed samples, and the other contains all odd-indexed samples. The formulas used in subsequent sections will leverage this separation. The main contribution and novelty of this work lie in achieving increased processor efficiency and reduced resource usage. This is achieved by proposing a hardware design of FFT architectures that utilizes a new parallel prefix adder. Compared to existing architectures, the designed FFT architecture offers lower power consumption, smaller chip area and faster operation speed [16, 17].

In FFT processors, Radix-2, Radix-4 and Split-Radix algorithms are commonly used to handle complex arithmetic operations. These algorithms involve various arithmetic operations, including addition and multiplication. Addition can be implemented in two ways: serial or parallel. Serial adders, such as full adders (FAs) and ripple carry adders (RCAs), are slow due to their bit-by-bit processing, have higher power consumption, and require more time for implementation. In contrast, parallel adders are significantly faster because they add multiple bits simultaneously. Designing high-speed, low-power parallel prefix (PP) adders is crucial for efficient FFT implementations. PP adders, such as

Kogge and Stone (KS) [18], Brent and Kung (BK) [19], Ladner and Fischer (LF) [20], and Han and Carlson (HC) [21], are popular choices due to their efficiency. Notably, parallel prefix adders offer a key advantage over other adder architectures. They can calculate the carry bits for all stages simultaneously, leading to faster operation. While the reference [22] explores using a Brent–Kung speculative adder for high-speed applications, this approach requires a larger area for processing due to its more complex design, potentially increasing hardware complexity.

1.1 Motivations and contributions

Traditional multiplication and addition circuits often lack optimization in terms of area usage, power consumption and processing speed. Developing high-speed FFT processors necessitates more efficient arithmetic units. This research proposes well-designed circuits that use an efficient adder to achieve improved performance. The key innovation lies in implementing Split-Radix, Radix-2, and Radix-4 FFT structures with a novel parallel prefix (PP) adder. This approach offers significant advantages over existing FFT designs using PP adders. It reduces power consumption and hardware footprint (area) while simultaneously enhancing processing speed. Furthermore, the paper presents a methodology for deploying adders on field-programmable gate arrays (FPGAs). This methodology provides a foundational building block for various DSP filters. The performance of the proposed design is evaluated based on key metrics, i.e., power consumption, delay, operating frequency, number of lookup tables (LUTs) utilized, and power-delay product. We first implemented four different conventional parallel prefix adders. To address limitations in existing designs, we then introduced a novel PP adder that achieves increased speed and lower power consumption. This novel PP adder was subsequently used to construct Split-Radix, Radix-2, and Radix-4 FFT processors. The performance of these processors was evaluated based on area, delay, and power consumption.

In this paper, we propose a novel parallel prefix adder for implementing Radix-2, Radix-4, and Split-Radix FFT algorithms. We conducted a performance analysis based on power consumption, speed, operating frequency, and resource utilization for systems using conventional adders and our proposed arithmetic structure. Simulation results demonstrate that the new parallel prefix adder exhibits impressive performance compared to these conventional structures, particularly in terms of lower power consumption and higher speed. This makes it an ideal candidate for DSP applications where high processing speed is essential. By enhancing the performance of adder blocks, we can minimize resource utilization and consequently reduce the number of computational stages required in different FFT structures. All conventional adders used for comparison were implemented using Xilinx software on an Intel core processor. Our proposed adder offers several advantages such as optimized area, high speed and lower power consumption. Notably, it achieves a power-delay product of 0.380 Wns, making it a compelling choice for designing various blocks in signal processing applications.

1.2 Organizations

The remainder of the work is organized as follows. Section 2 describes the implementation of the FFT structures, including both conventional and proposed arithmetic structures. Section 3 presents the findings from the simulations. Finally, Sect. 4 summarizes the key takeaways of the paper.

2 Methodology

In this paper, we explore the potential of various FFT algorithms, including Split-Radix, Radix-4, and Radix-2. While the Radix-2 algorithm is commonly used in FFT processors to handle complex number arithmetic operations, i.e., addition and multiplication, all three algorithms can be implemented with efficient adders. Traditional multiplication and addition circuits often lack optimization in terms of area usage, power consumption and processing speed. However, high-speed FFT processors necessitate more efficient arithmetic units. This research addresses this challenge by introducing well-designed circuits that use a novel parallel prefix adder to achieve improved performance. The key innovation lies in implementing Split-Radix, Radix-2 and Radix-4 FFT structures with this novel adder. This approach offers significant advantages over existing FFT designs using PP adders. It simultaneously reduces power consumption, minimizes hardware footprint (area), and enhances processing speed.

2.1 Radix-2 butterfly algorithm

In DFT, when M is a power of 2, we can separate the even and odd points in the DFT formula. As a result, the common DFT equation converts to the following formulas,

$$Y_{2k} = \sum_{n=0}^{\frac{M}{2}-1} \left(y_n + y_{n+M/2}\right) \cdot W_{M/2}^{kn}, \tag{1}$$

$$Y_{2k+1} = \sum_{n=0}^{\frac{M}{2}-1} \left(y_n - y_{n+M/2}\right).W_{M/2}^{n} \cdot W_{M/2}^{kn}, \tag{2}$$

In a radix-2 FFT, the input data are typically arranged in a sequential order. However, the output data after the FFT computation exhibits a bit-reversed order. Hardware implementations often require rearranging the bit-reversed output back to a sequential format for further processing. Figure 1 illustrates a 32-point radix-2 FFT structure. It

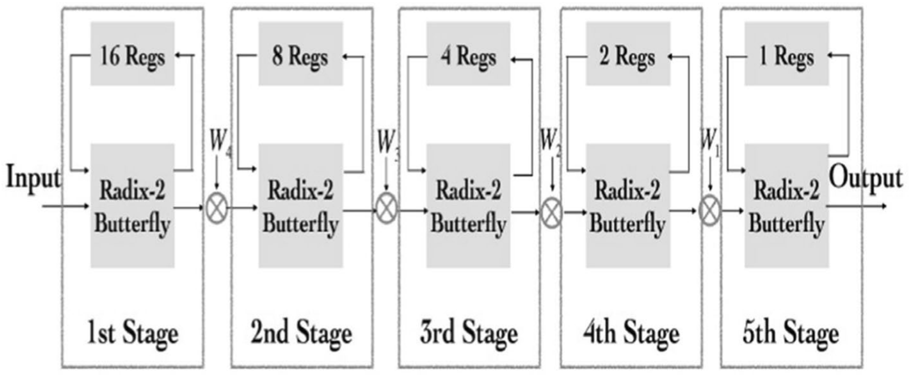

Fig. 1 Different parts of 32-point FFT structure

consists of multiple stages, each containing registers and a radix-2 butterfly unit. The number of registers in each stage depends on the size of the input data. Between consecutive stages, the outputs are multiplied by complex twiddle factors [16, 23].

The Radix-2 butterfly structure significantly reduces the number of multipliers and adders required for the FFT computation. For instance, without the Radix-2 FFT, the first stage necessitates 16 multipliers and 32 adders. However, using the Radix-2 FFT with the butterfly structure, only 1 multiplier and 2 adders are needed in each stage. It is important to note that the number of registers remains the same. In this work, the Radix-2 FFT implementation uses the decimation-in-time (DIT) algorithm. DIT algorithms process data in the time domain first, with the input being bit-reversed and the output in natural order. In contrast, decimation-in-frequency (DIF) algorithms process data in the frequency domain first, having the input in natural order and the output bit-reversed. The Radix-2 FFT structure is a core component of the FFT, enabling efficient computation of complex number operations within the FFT algorithm.

The synthesis output of the FFT simulation suggests that employing Radix-2 structure for a 32-point FFT would be efficient in terms of speed and area [13, 14]. The flow diagram of the Radix-2 FFT structure is depicted in Fig. 2. The multiplier block is succeeded by a PP adder block. The final output is determined by the value of C_{in}. If $C_{in} = 1$, then the equal real and imaginary parts are computed. Otherwise, unequal real and imaginary parts are calculated.

2.2 Radix-4 butterfly algorithm

In DFT, when M is a power of 4, we can convert Y_{2k} to Y_{4k} and Y_{4k+2} as well as Y_{2k+1} to Y_{4k+1} and Y_{4k+3} so that the following formulas are obtained [14]:

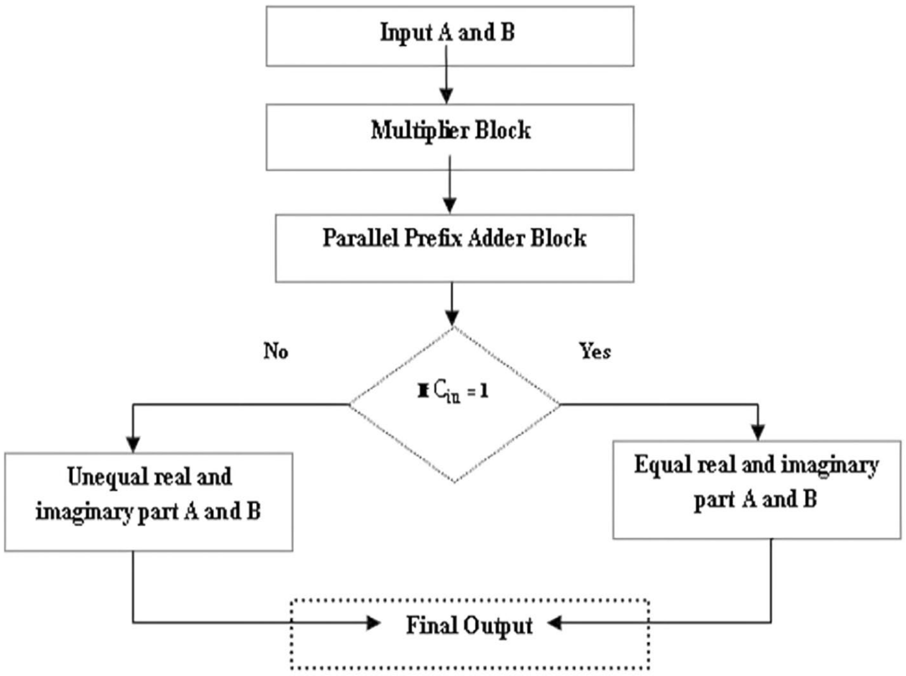
Input A and B
Multiplier Block
Parallel Prefix Adder Block
No
Yes
If $C_{in} = 1$
Unequal real and imaginary part A and B
Equal real and imaginary part A and B
Final Output

Fig. 2 Flow diagram of Radix-2 FFT structure

$$Y_{4k} = \sum_{n=0}^{\frac{M}{4}-1} \left(Y_n + Y_{n+M/4} + Y_{n+M/2} + Y_{n+3M/4}\right) \cdot W_M^0 \cdot W_{M/4}^{kn}, \tag{3}$$

$$Y_{4k+1} = \sum_{n=0}^{\frac{M}{4}-1} \left(Y_n - Y_{n+\frac{M}{4}} - Y_{n+M/2} + Y_{n+3M/4}\right) \cdot W_M^n \cdot W_{M/4}^{kn}, \tag{4}$$

$$Y_{4k+2} = \sum_{n=0}^{\frac{M}{4}-1} \left(Y_n - Y_{n+\frac{M}{4}} + Y_{n+\frac{M}{2}} - Y_{n+\frac{3M}{4}}\right) \cdot W_M^{2n} \cdot W_{\frac{M}{4}}^{kn}, \tag{5}$$

$$Y_{4k+3} = \sum_{n=0}^{\frac{M}{4}-1} \left(Y_n + Y_{n+\frac{M}{4}} - Y_{n+\frac{M}{2}} - Y_{n+\frac{3M}{4}}\right) \cdot W_M^{3n} \cdot W_{\frac{M}{4}}^{kn}. \tag{6}$$

The graph obtained from this method for Radix-4 DIF-FFT with 16-point is shown in Fig. 3 with the conversion size $M = 16$ [24, 25].

2.3 Split-Radix butterfly algorithm

This method is a combination of two aforementioned methods so that only in the third formula, we change Y_{2k+1} to Y_{4k+1} and Y_{4k+3} so that the following formulas are obtained. The structure of Split-Radix 3/6 FFT by block diagram is shown in Fig. 4.

$$Y_{2k+1} = \sum_{n=0}^{\frac{M}{2}-1} \left(Y_n - Y_{n+M/2}\right) \cdot W_{M/2}^n \cdot W_{M/2}^{kn}, \tag{7}$$

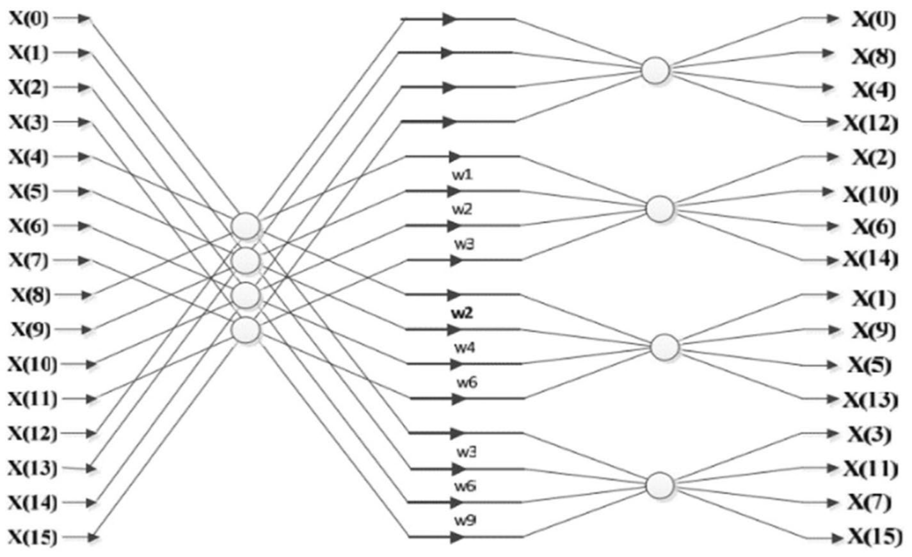

Fig. 3 Structure of 16-point Radix-4 FFT by signal flow graph

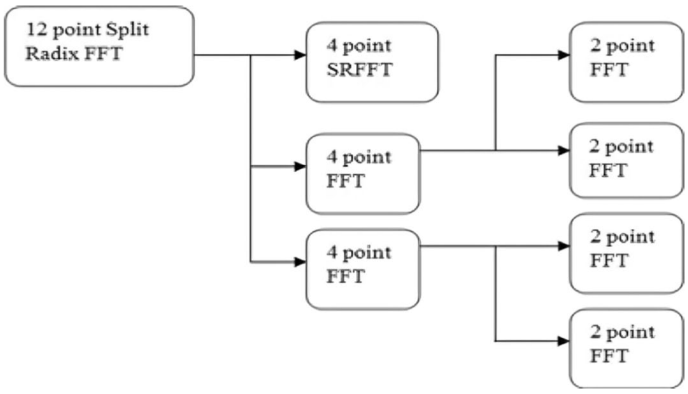

Fig. 4 Structure of Split-Radix FFT by block diagram

$$Y_{4k+1} = \sum_{n=0}^{\frac{M}{4}-1} \left(Y_n - jY_{n+\frac{M}{4}} - Y_{n+M/2} + jY_{n+3M/4}\right) \cdot W_M^n \cdot W_{M/4}^{kn}, \tag{8}$$

$$Y_{4k+3} = \sum_{n=0}^{\frac{M}{4}-1} \left(Y_n + jY_{n+M/4} - Y_{n+\frac{M}{2}} - jY_{n+3M/4}\right) \cdot W_M^{3n} \cdot W_{M/4}^{kn}. \tag{9}$$

Split-Radix 3/6 FFT approach is defined based on the following equations:

$$Y(k) = A_k + W_2^k r W_3^k m B_k + C_k + W_3^{-k} m E_k + W_2^{-k} r W_3^{-k} m F_k, \tag{10}$$

$$A_k = \sum_{n=0}^{\frac{M}{3}-1} x_{3n} W_{\frac{M}{3}}^{nk}, \tag{11}$$

$$B_k = \sum_{n=0}^{\frac{M}{6}-1} x_{6n} + 2^r + 3^m W_{\frac{M}{6}}^{nk}, \tag{12}$$

$$C_k = \sum_{n=0}^{\frac{M}{6}-1} x_{6n} + 2^r W_{\frac{M}{6}}^{nk}, \tag{13}$$

$$E_k = \sum_{n=0}^{\frac{M}{6}-1} x_{6n} - 2^r W_{\frac{M}{6}}^{nk}, \tag{14}$$

$$F_k = \sum_{n=0}^{\frac{M}{6}-1} x_{6n} - 2^r - 3^m W_{\frac{M}{6}}^{nk}. \tag{15}$$

The structure of Split-Radix FFT by block diagram is given in Fig. 4. If the range of n is considered from 0 to $\frac{N}{6}$ in the aforementioned equations, $Y(k)$ will be completely obtained. The design of the suggested Split-Radix FFT is summarized. The primary input array with a length equal to M is divided into five arrays. This process for each of new sub-series is successively repeated until the size of all sub-DFT's would not be divisible by 6 [26, 27].

We consider the performance of the suggested structure by investigating the computational complexity. Moreover, it will be compared with the other structures. We suppose that AN and MN are the number of additions and multiplications, respectively. Moreover, a 3-point DFT needs 12 real additions and 4 real multiplications [28, 29].

2.4 Conventional PP adders

PP adders significantly outperform serial adders by performing addition on multiple bits simultaneously. They offer a good theoretical foundation for balancing power consumption, speed, and occupied area in circuit design, making them ideal for designing high-performance integrated circuits due to their balance between speed, power, and area consumption (VLSI synthesis). The flow diagram of a 16-bit PP adder often includes three stages, i.e., pre-computation, prefix network, and post-computation. Each stage performs specific operations to calculate generate, propagate, carry and sum signals. There are various types of PP adders, including Kogge–Stone (KS), Brent–Kung (BK), Ladner–Fischer (LF), and Han–Carlson (HC) adders. One of the most widely used and significant adders is the Kogge–Stone parallel prefix adder (KSPP), illustrated in Fig. 5. The KSPP comprises 5 stages. In stage 1, the pre-processing stage calculates the generate (g_i) and propagate (p_i) values. Stages 2 to 4 form a prefix network where the final carry (c_i) is determined based on the g_i and p_i values from the previous stage, and the sum (s_i)

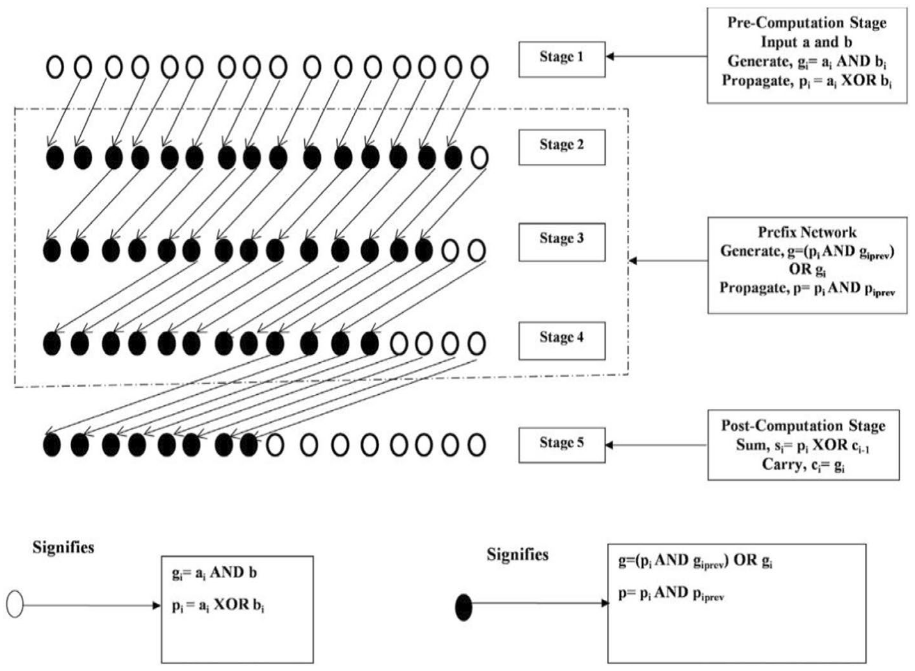
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Pre-Computation Stage
Input a and b
Generate, $g_i = a_i$ AND b_i
Propagate, $p_i = a_i$ XOR b_i
Prefix Network
Generate, $g=(p_i$ AND $g_{iprev})$ OR g_i
Propagate, $p = p_i$ AND p_{iprev}
Post-Computation Stage
Sum, $s_i = p_i$ XOR c_{i-1}
Carry, $c_i = g_i$
Signifies
$g_i = a_i$ AND b
$p_i = a_i$ XOR b_i
Signifies
$g=(p_i$ AND $g_{iprev})$ OR g_i
$p = p_i$ AND p_{iprev}

Fig. 5 Different stages of KSPP adder

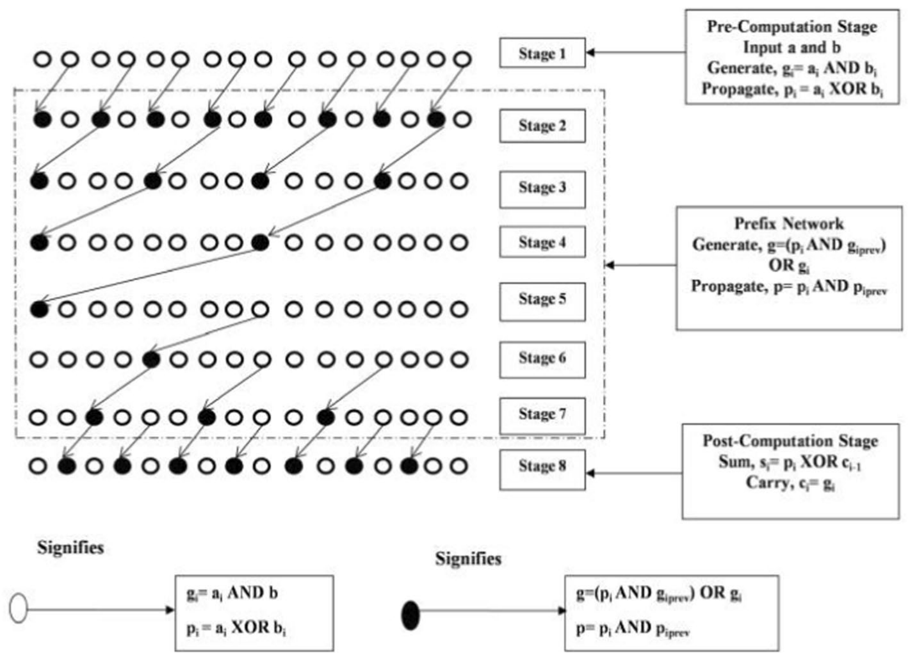

Fig. 6 Different stages of BKPP adder

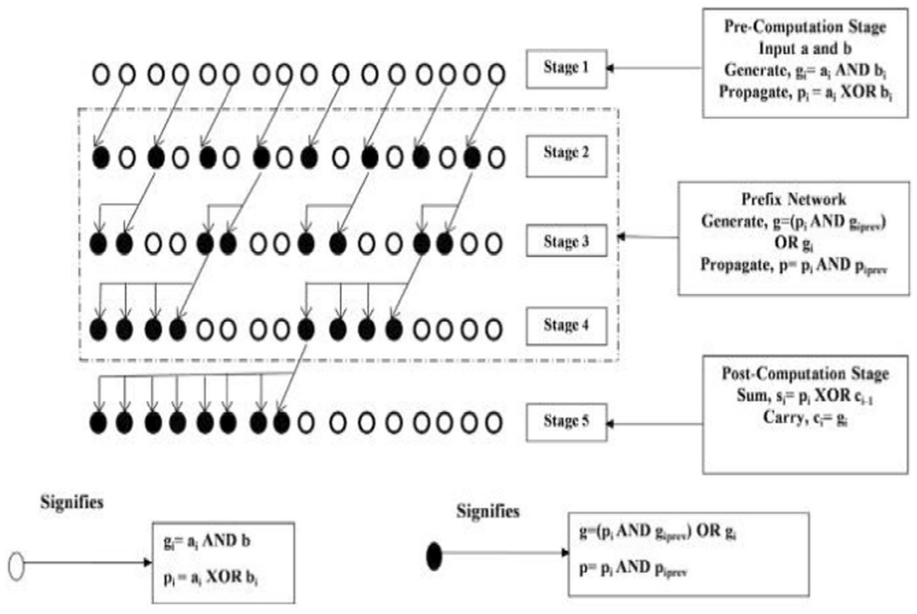
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Pre-Computation Stage
Input a and b
Generate, g_i= a_i AND b_i
Propagate, p_i = a_i XOR b_i
Prefix Network
Generate, g=(p_i AND g_{iprev}) OR g_i
Propagate, p= p_i AND p_{iprev}
Post-Computation Stage
Sum, s_i= p_i XOR c_{i-1}
Carry, c_i= g_i
Signifies
g_i= a_i AND b
p_i = a_i XOR b_i
Signifies
g=(p_i AND g_{iprev}) OR g_i
p= p_i AND p_{iprev}

Fig. 7 Different stages of LFPP adder

is calculated in the final stage. Moreover, BKPP, LFPP and HCPP adders are designed as shown in Figs. 6, 7 and 8, respectively.

As observed from Fig. 6, the BKPP adder comprises 8 stages. The first stage serves as a pre-processing stage, calculating generate (g_i) and propagate (p_i) values for each bit position. Stages 2 to 7 form a prefix network that utilizes these values to determine the final carry for each bit. This approach is similar to the Ladner–Fischer Parallel Prefix

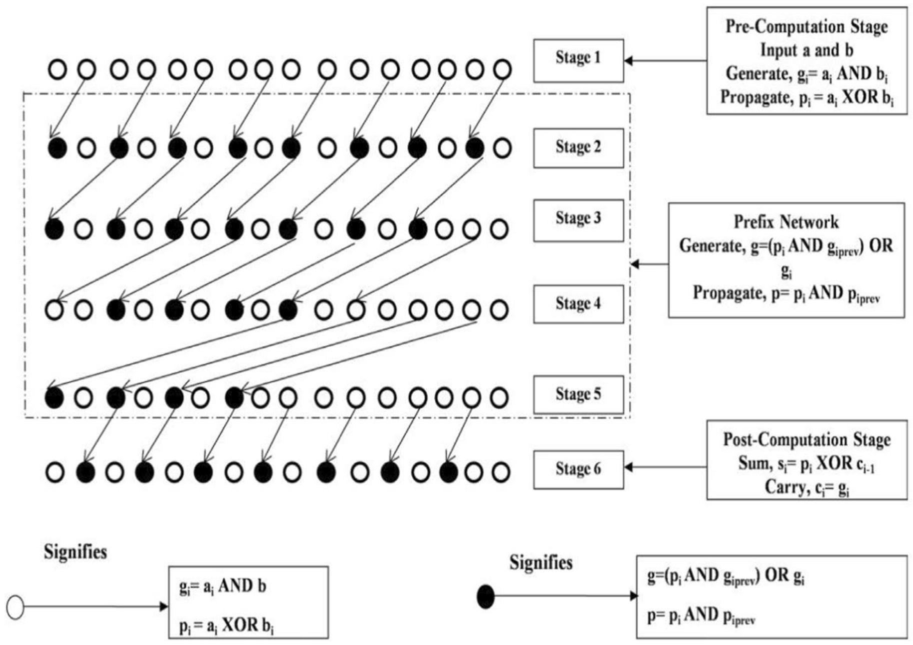

Fig. 8 Different stages of HCPP adder

(LFPP) adder in Fig. 7. In the LFPP adder, stage 1 also performs the pre-processing to calculate g_i and p_i values. Stages 2–4 form the prefix network stage, and the final stage computes the sum for each bit position.

The Han–Carlson parallel prefix (HCPP) adder in Fig. 8 is a hybrid design that combines elements from both the BKPP and KSPP adders. Similar to the previous adders, stage 1 performs the pre-processing for g_i and p_i values. Stages 2–5 form the prefix network, and the last stage calculates the final sum. Interestingly, the HCPP structure shares stages 1 and 6 with the BKPP adder; while, stages 2–5 have the same structure as the KSPP adder. This hybrid approach offers potential benefits in terms of lower power consumption and smaller area footprint on the chip due to reduced wiring connections and gates, but it introduces an additional stage compared to the BKPP and LFPP adders.

2.5 Implementation of Radix-2, Radix-4 and Split-Radix using proposed arithmetic structures

In this paper, we introduce a novel adder design that, when combined with the previously mentioned FFT structures, significantly reduces the number of multipliers and adders required for FFT computations. For instance, without the Radix-2 FFT, the first stage in Fig. 1 would necessitate 16 multipliers and 32 adders. However, with the implementation of the radix-2 FFT, only 1 multiplier and 2 adders are needed in each stage. It is important to note that the number of registers remains unchanged. This reduction in multipliers and adders translates to significant area savings. Additionally, by employing parallel prefix adders, we achieve improved performance compared to traditional serial adders. Parallel prefix adders excel due to their ability to perform addition on multiple bits simultaneously, unlike serial adders which process bits one after another.

The proposed architecture depicted in Fig. 9 offers reduced complexity compared to conventional PP adders. Similar to other parallel prefix adders, Stage 1 calculates the generate and propagate values. Stages 2–4 utilize these values to compute the final carry, and the sum is determined in the last stage. The new parallel prefix adder exhibits improved performance and efficiency compared to previously reported adders. This makes it particularly attractive for applications where speed is a critical factor. The proposed adder has been simulated, and the detailed results are presented in the next section.

3 Simulation results

The simulations for this work were conducted using Xilinx Vivado and ISE. The M-point input array was converted to its 16-bit binary equivalents by applying twiddle factors and performing computations within the input block. Simulations were performed for Radix-2, Radix-4 and Split-Radix FFT algorithms. The new arithmetic structure demonstrates significant advantages, particularly in terms of power consumption, number of LUTs, operating frequency and delay compared to other structures. This makes it highly suitable for DSP applications where high processing speed is essential. By enhancing the performance of adder blocks, we can minimize resource utilization and consequently reduce the number of computational stages required in FFT implementations. Table 1 summarizes the performance of various adders using metrics such as area, speed, power consumption and power-delay product. As shown in Table 1, the HCPP adder exhibits a lower delay (12.147 ns) compared to KSPP and LFPP adders. This delay is further broken down into logic delay (5.417 ns) and router delay (6.73 ns). Additionally, the HCPP adder utilizes only 24 LUTs, demonstrating a more compact design compared to the KSPP (38 LUTs), BKPP (33 LUTs), and LFPP

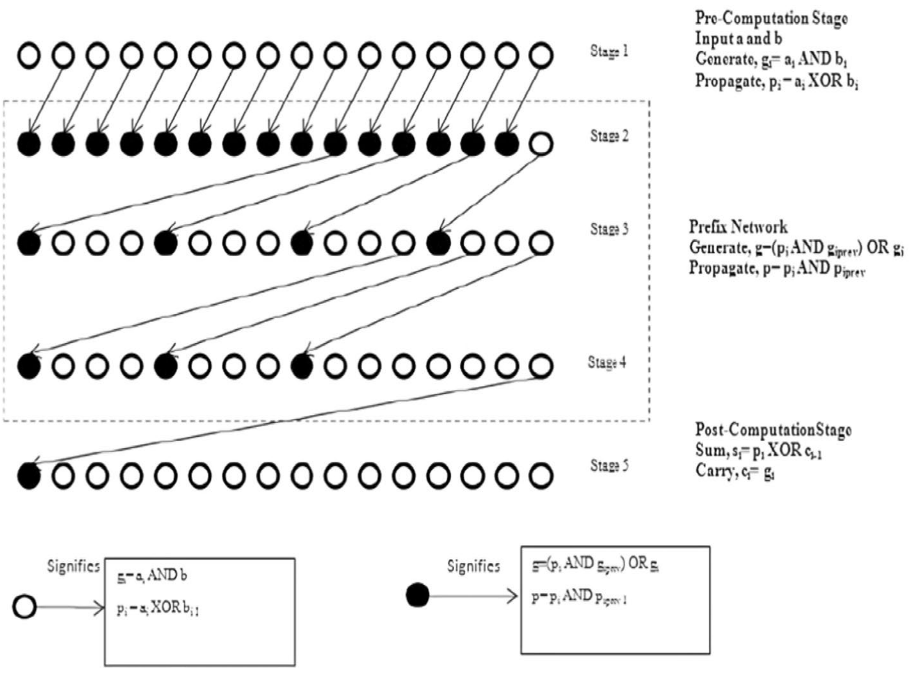
Stage 1
Pre-Computation Stage
Input a and b
Generate, $g_i = a_i$ AND b_i
Propagate, $p_i = a_i$ XOR b_i
Stage 2
Stage 3
Prefix Network
Generate, $g=(p_i$ AND $g_{iprev})$ OR g_i
Propagate, $p = p_i$ AND p_{iprev}
Stage 4
Stage 5
Post-ComputationStage
Sum, $s_i = p_i$ XOR c_{i-1}
Carry, $c_i = g_i$
Signifies
$g_i = a_i$ AND b
$p_i = a_i$ XOR b_{i1}
Signifies
$g=(p_i$ AND $g_{iprev})$ OR g_i
$p = p_i$ AND p_{iprev1}
Fig. 9 Our proposed adder

Table 1 Comparison of our proposed adder with the conventional adders

	Number of bonded IOB		Delay (ns)		Number of slice LUTs	Total power (W)		Power delay product
	I-Buf	O-Buf	Logic	Router		Leakage	IOs	
KSPP	32	17	6.955	5.417	38	0.064	0.017	0.969
BKPP	33	17	6.070	5.159	33	0.064	0.017	0.909
LFPP	33	17	6.931	5.417	26	0.064	0.017	1.000
HCPP	32	17	6.730	5.417	24	0.064	0.017	0.983
Our proposed adder	32	17	4.609	3.852	23	0.029	0.017	0.380

Table 2 Device utilization summary for 16 point FFT using Radix-2 Algorithm

Device utilization summary (estimated values)			(·)
Logic utilization	**Used**	**Available**	**Utilization (%)**
Number of slice registers	503,020	708,480	71
Number of slice LUTs	297,561	354,240	84
Number of fully used LUT · FF pairs	152,704	254,508	60
Timing summary			
Speed grade: — 2			
Minimum period: 2.977 ns (maximum frequency: 335.914 MHz)			
Minimum input arrival time before clock: 7.304 ns			
Minimum output required time after clock: 0.659 ns			
Minimum combinational path delay: no path found			

It describes the logic utilization based on the number of slice registers, the number of slice LUTs and the number of fully used LUT-FF pairs

(26 LUTs) adders. While the BKPP adder has a lower delay than other adders, it occupies a large area based on the number of LUTs. Our proposed adder offers a compelling combination of optimized area, high speed, and lower power consumption (with a power-delay product of 0.380 Wns), making it a preferred choice for designing various blocks in signal processing applications.

We present the results of the synthesis, focusing on the number of utilized LUTs. Tables 2, 3, and 4 present the device utilization summary for different FFT structures when our proposed adder is used. These tables show the utilization of slice LUTs for Radix-2, Radix-4, and Split-Radix structures, respectively. As observed in the tables, the Split-Radix structure exhibits the lowest LUT utilization at 4%, followed by Radix-4 at 71%, and Radix-2 at 84%. Tables 5, 6, and 7 present the number of calculation operations (multiplications and additions) required for different FFT structures with sizes $M = 64$, $M = 256$ and $M = 1024$, respectively. As expected, the number of multiplications and additions increases for all structures as the data size (M) increases. However, the tables also reveal that the Split-Radix structure consistently requires fewer multiplications and additions compared to Radix-2 and Radix-4 structures for all data sizes evaluated.

We have compared the processing power, the number of LUTs, working frequency and delay for different algorithms in Figs. 10, 11, 12 and 13 for chips using

Table 3 Device utilization summary for 16 point FFT using Radix-4 Algorithm

Device utilization summary (estimated values)			**(·)**
Logic utilization	**Used**	**Available**	**Utilization (%)**
Number of slice registers	123,406	708,480	17
Number of slice LUTs	213,700	354,240	60
Number of fully used LUT · FF pairs	82,598	254,508	32
Timing summary			
Speed grade: — 3			
Minimum period: 2.652 ns (maximum frequency: 365.914 MHz) Minimum input arrival time before clock: 7.304 ns			
Minimum output required time after clock: 0.350 ns			
Minimum combinational path delay: no path found			

It describes the logic utilization based on the number of slice registers, the number of slice LUTS and the number of fully used LUT-FF pairs

Table 4 Device utilization summary for FFT using Split-Radix algorithm

Device utilization summary (estimated values)			
Logic utilization	**Used**	**Available**	**Utilization (%)**
Number of slice registers	28,339	708,480	4
Number of slice LUTs	35,424	354,240	10
Number of fully used LUT · FF pairs	203,606	254,508	8
Timing summary			
Speed grade: — 4			
Minimum period: 2.105 ns (maximum frequency: 450.914 MHz) Minimum input arrival time before clock: 7.304 ns			
Minimum output required time after clock: 0.150 ns			
Minimum combinational path delay: no path found			

It describes the logic utilization based on the number of slice registers, the number of slice LUTS and the number of fully used LUT-FF pairs

Table 5 The number of sums and multiplications for $M = 64$

FFT	**Radix-2**	**Radix-4**	**Split-Radix**
The number of multiplication	462	208	196
The number of sum	1321	976	964

Table 6 Number of sums and multiplications for $M = 256$

FFT	**Radix-2**	**Radix-4**	**Split-Radix**
The number of multiplication	1800	1392	1284
The number of sum	5896	5488	5380

Table 7 Number of sums and multiplications for $M = 1024$

FFT	**Radix-2**	**Radix-4**	**Split-Radix**
The number of multiplication	10,248	7856	7172
The number of sum	30,728	28,336	27,652

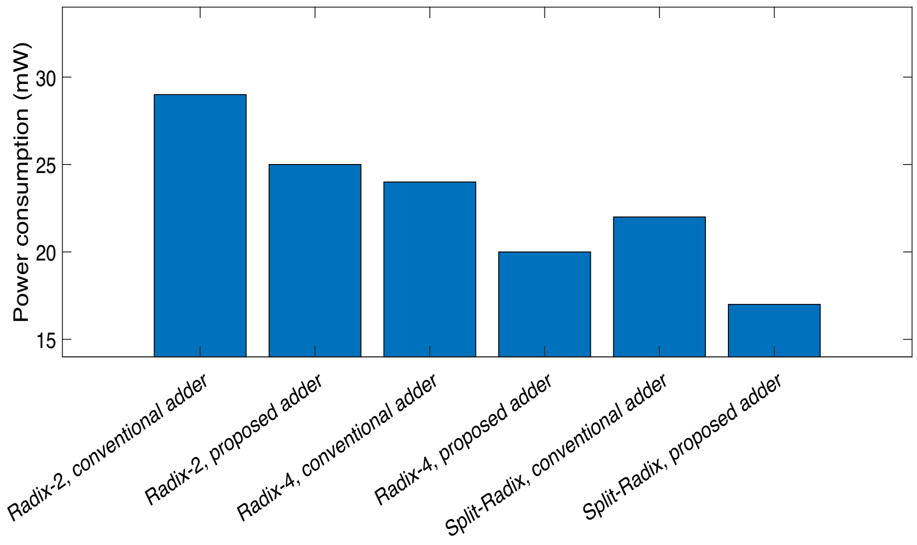

Fig. 10 The power consumption (mW) for different adders and algorithms

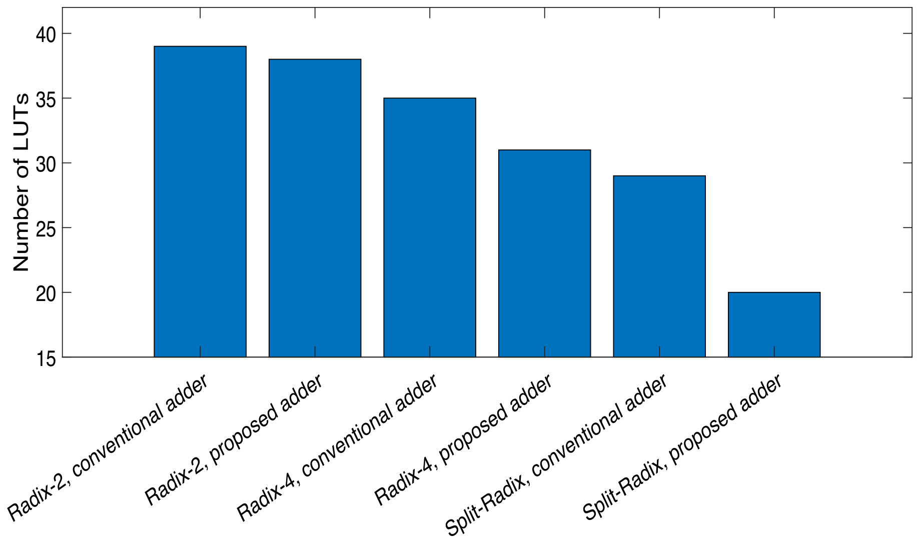

Fig. 11 Number of LUTs for different adders and algorithms

conventional adders and our proposed adder. According to the values obtained from the simulations in Fig. 10, the power consumption of the Split-Radix algorithm is less than the other methods which is 17 mW and 22 mW for the circuit using proposed adder and conventional adders, respectively which this means that the system is more optimized. Moreover, in the case of using our proposed adder for all FFT algorithms, the consumed power is less than those of conventional adders. As expected, Split-Radix FFT deemed more suitable for FPGA implementation than the Radix-4 and Radix-2 algorithms and it is well-suited for a low-power FFT processor because it requires the fewest arithmetic operations among all FFT algorithms.

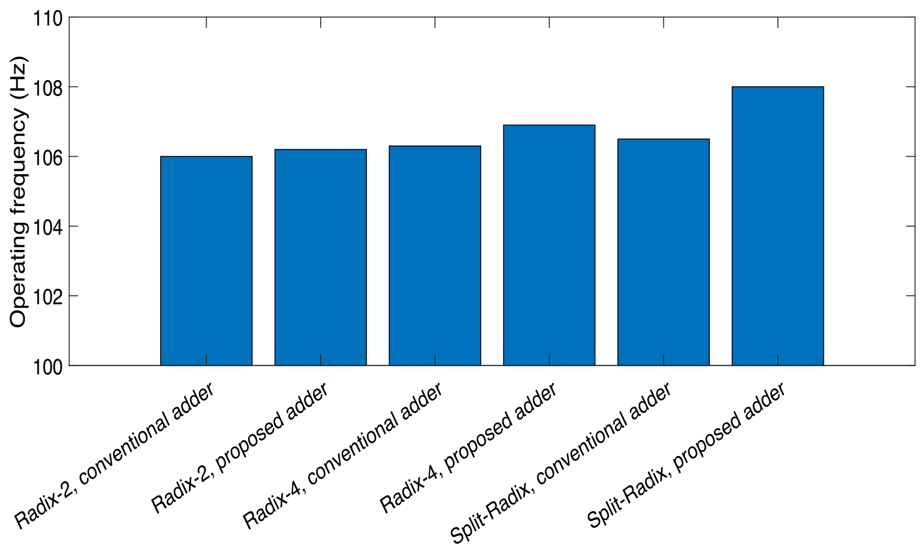

Fig. 12 Operating frequency (Hz) of the system for different adders and algorithms

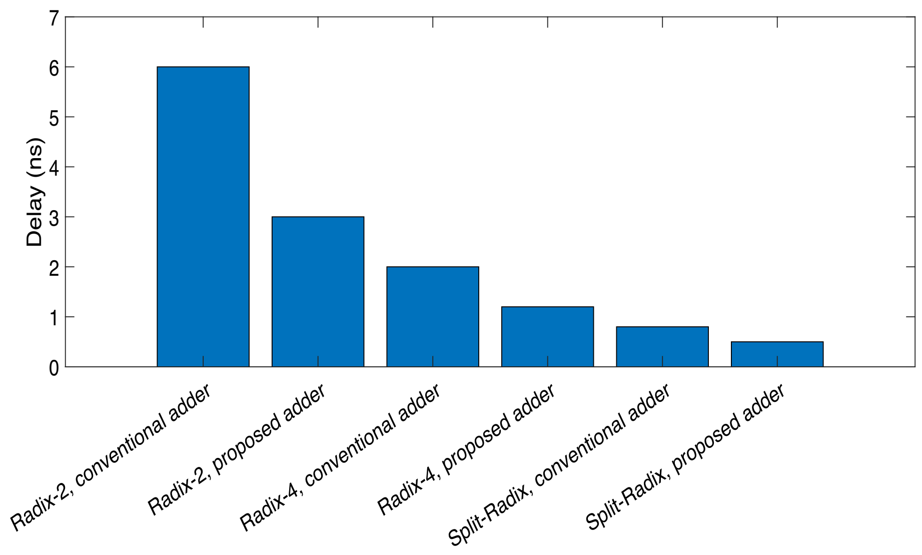

Fig. 13 Delay (ns) of the system for different adders and algorithms

Figure 11 demonstrates that using our proposed adder leads to lower resource utilization compared to conventional adders. For instance, the Split-Radix algorithm with the proposed adder utilizes fewer resources than the Radix-4 algorithm with a conventional adder. The chips are made up of many LUTs, and the less the number of LUTs used, the more resources are available for the system to perform other calculations. As a result, it increases the speed of the system which can be seen in the Split-Radix method [27]. Operating frequency in Fig. 12 refers to the number of cycles a processor can execute per second. A higher operating frequency generally translates to faster execution of instructions. Based on the results, the Split-Radix algorithm achieves a higher operating frequency, indicating faster operation.

Figure 13 shows the delay experienced by different FFT algorithms implemented with both the conventional adder and the proposed adder. The results indicate that the proposed adder offers lower delay compared to conventional adders for all algorithms. Additionally, the Split-Radix algorithm exhibits the lowest delay among all algorithms based on this metric. Specifically, the delays for Radix-2, Radix-4, and Split-Radix with the proposed adder are 3 ns, 1.2 ns, and 0.5 ns, respectively.

4 Conclusion

In this paper, we have proposed a new arithmetic structure. The aforementioned structure demonstrates impressive performance compared to other structures. It is particularly suitable for use in DSP applications where high processing speed is crucial. By enhancing the performance of adder blocks, the resource utilization and number of computational stages in the implementation of FFT algorithms can be minimized. Our proposed adder offers optimized area, high speed and lower power consumption, making it a preferred choice for designing various blocks in signal processing applications. According to the results of device utilization and computational complexity, Radix-4 and Split-Radix methods are better than Radix-2 method. In the simulation results section, the power consumption, the used chip surface, the working frequency, and delay in the aforementioned methods are compared. From comparing the results, we can see that Radix-4 and Split-Radix are better than Radix-2 algorithm and they work more efficiently. Finally, according to the changes applied in the Split-Radix algorithm, it has a very high efficiency, which is suitable for complex applications.

Author contributions

Mahsa Shirzadian Gilan proposed the main idea and analyzed the result. She performed the simulations, evaluation, methodology and editing the paper. Behrouz Maham edited and evaluated the paper. All authors read and approved the final manuscript.

Funding

This research is supported by the Ministry of Science and Higher Education of the Republic of Kazakhstan, under the project AP13068587 entitled "57–65 GHz Wireless Communication Front-End for Secure 5 G Applications".

Declaration

Competing interests

The authors declare that they have no competing interests.

Received: 16 May 2024 Accepted: 14 August 2024
Published online: 24 August 2024

References

1. Y.S. Algnabi, R. Teymourzadeh, M. Othman, FPGA implementation of pipeline digit-slicing multiplier-less radix 2^2 DIF SDF butterfly for fast Fourier transform structure (IMEN) (2018). CoRR arXiv:1806.04570
2. S.J. Saenz, J.J. Raygoza, E.C. Becerra, FPGA design and implementation of Radix-2 fast Fourier transform algorithm with 16 and 32 points, in *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). Ixtapa, Mexico*, vol. 2015 (2015), 1–6
3. M.S. Gilan, B. Maham, Diversity achieving full-duplex DF relaying with joint relay-antenna selection under Nakagami-m fading environment. Int. J. Commun. Syst. **36**(5), e5427 (2023)
4. M. Schlemon, J. Naghmouchi, FFT optimizations and performance assessment targeted towards satellite and airborne radar processing, in *IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). Porto, Portugal*, vol. 2020 (2020), pp. 313–320
5. L. Santhosh, A. Thomas, Implementation of radix 2 and radix 2^2 FFT algorithms on Spartan6 FPGA, in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, India* (2013), pp. 1–4

6. M.S. Gilan, B. Maham, Performance analysis of power-efficient IRS-assisted full duplex NOMA systems. Phys. Commun. **64**, 102338 (2024)
7. E. Guariglia, R.C. Guido, Chebyshev wavelet analysis. J. Funct. Spaces **1**, 5542054 (2020)
8. M.V. Berry, Z.V. Lewis, On the Weierstrass–Mandelbrot fractal function. Proc. R. Soc. Lond. Ser. A Math. Phys. Sci. **370**(1743), 459–484 (1980)
9. Y. Wang, S. Cui, Hyperspectral image feature classification using stationary wavelet transform, in *2014 International Conference on Wavelet Analysis and Pattern Recognition* (2014), pp. 104–108
10. E. Guariglia, Harmonic Sierpinski gasket and applications. Entropy **20**, 714 (2018)
11. X. Zheng, Y.Y. Tang, J. Zhou, A framework of adaptive multiscale wavelet decomposition for signals on undirected graphs. IEEE Trans. Signal Process. **67**(7), 1696–1711 (2019)
12. E. Guariglia, S. Silvestrov, *Fractional-Wavelet Analysis of Positive Definite Distributions and Wavelets on D′(C)* Engineering Mathematics II. (Springer, Berlin, 2016), pp.337–353
13. K.K. Parthi, *VLSI Digital Signal Processing Systems* (Wiley, Hoboken, 1999)
14. Z. Ying-Xi, S. Lei, Design of mixed-radix FFT algorithm based on FPGA, in *2022 7th International Conference on Communication, Image and Signal Processing (CCISP), Chengdu, China* (2022), pp. 418–422
15. M.S. Gilan, R. Paranjape, Beam tracking in phased array antenna based on the trajectory classification. Trans. Emerg. Telecommun. Technol. **34**(6), e4769 (2023)
16. G. Akkad, A. Mansour, B. ElHassan, F.L. Roy, M. Najem, FFT Radix-2 and Radix-4 FPGA acceleration techniques using HLS and HDL for digital communication systems, in *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), Beirut, Lebanon* (2018), pp. 1–5
17. W.-S. Gan, A. Seth, S.M. Kuo, Versatile and portable DSP platform for learning embedded signal processing, in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic* (2011), pp. 2888–2891
18. P.M. Kogge, H.S. Stone, A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE Trans. Comput. **22**(8), 786–793 (1973)
19. R.P. Brent, H.T. Kung, A regular layout for parallel adders. IEEE Trans. Comput. **31**(3), 260–264 (1982)
20. R.E. Ladner, M.J. Fisher, Parallel prefix computation. J. Assoc. Comput. Mach. **27**(4), 831–838 (1980)
21. T. Hans, D.A. Carlson, Fast area-efficient VLSI adders, in *Proceedings of the 8th IEEE Symposium on Computer Arithmetic* (1987), pp. 49–56
22. G. Thakur, H. Sohal, S. Jain, FPGA-based parallel prefix speculative adder for fast computation application, in *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (2020), pp. 206–210
23. P.T.L. Pereira et al., Energy-quality scalable design space exploration of approximate FFT hardware architectures. IEEE Trans. Circuits Syst. I Regul. Pap. **69**(11), 4524–4534 (2022)
24. A. Sankaran, M.S. Reddy, K. Arunkumar, M. Bhaskar, Design and implementation of 1024 point pipelined radix 4 FFT processor on FPGA for biomedical signal processing applications, in *IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS). Chennai, India*, vol. 2020 (2020), pp. 1–6
25. E. Delaye, A. Sirasao, C. Dudha, S. Das, Deep learning challenges and solutions with Xilinx FPGAs, in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, USA* (2017), pp. 908–913
26. S. Magar, S. Shen, G. Luikuo, M. Fleming, R. Aguilar, An application specific DSP chip set for 100 MHz data rates, in *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing, New York, NY, USA* (1988), pp. 1989–1992
27. S. Li, X. Yue, P. Wang, Research and FPGA implementation of security control system based on high alphabet modulation, in *2018 IEEE 4th International Conference on Computer and Communications (ICCC), Chengdu, China* (2018), pp. 2710–2714
28. S. Liu, et al., Memory-efficient architecture for accelerating generative networks on FPGA, in *2018 International Conference on Field-Programmable Technology (FPT), Naha, Japan* (2018), pp. 30–37
29. T. Siu, C.-W. Sham, F.C.M. Lau, Operating frequency improvement on FPGA implementation of a pipeline large-FFT processor, in *2017 19th International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea (South)* (2017), pp. 5–9

Publisher's Note