International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Area and Delay Minimization of Radix-2^k Feedforward FFT Architecture

A. Salai Kishwar Jahan [1], A. Indhumathi [2]

[1, 2]Mother Terasa College of Engineering & Technology, Illuppur, Pudukkottai, India

Abstract: ***The radix-2^2 was a milestone in the design of pipelined FFT hardware architectures. Later, radix-2 extended to radix-2^{16}.However, radix-2^{16} was only proposed for single path delay feedback (SFD) architectures, but not for feedforward, and also it called multi path delay commutator (MDC) .The radix-216 feedforward Fast Fourier Transform architecture (FFT). In feedforward architectures radix-2^{16} can be used for any number of parallel samples which is a power of two. Furthermore, both decimation in frequency (DIF) and decimation in time (DIT) decompositions can be used. In addition to this, the designs can achieve very high throughputs and reduce the spare complexity, which make them suitable for the most demanding applications. Indeed, the proposed radix-2^k feedforward architectures require fewer hardware resources than parallel feedback ones, also called multi path delay feedback (MDF), when several samples in parallel must be processed. As result, the proposed radix-2^{16} feedforward architectures not only offer an attractive solution for current applications, but also open up a new research line on feedforward structures.***

Keywords: Fast Fourier Transform, Multi path delay feedback (MDF), Pipelined Architecture.

1. Introduction

The Fast Fourier transform (FFT) is one of the most important algorithms in the field of digital signal processing. It is used to calculate the discrete Fourier transform (DFT) efficiently. These implementations can be mainly classified into memory-based and pipeline architecture style. Memory-based architecture is widely adopted to design, also known as the single Processing Element (PE) approach. This design style usually composed of a main PE and several memory units, thus the hardware cost and power consumption are both lower than the other architecture style. But disadvantage is that it has long latency, long throughput and it cannot be parallized. In order to meet the high performance and real-time requirements of modern applications, hardware designers have always tried to implement efficient architectures for the computation of the FFT.

For a pipelined FFT processor, each stage has its own set of processing elements. All the stages are computed as soon as data are available. Pipelined FFT processor have features like simplicity, modularity and high throughput low hardware complexity, and low power consumption. These features are important for real-time, in-place applications where the input data often arrive in a natural sequential order. We therefore select the pipeline architecture for our FFT processor implementation. Pipelined hardware architecture [9], because they provide high throughputs and low latencies suitable for real time, as well as a reasonably low area and low power consumption. There are two main types of pipelined architectures: feedback (FB) [14] and feed forward (FF) [3]. On the one hand, feedback architectures [14] are characterized by their feedback loops, i.e., some outputs of the butterflies are fed back to the memories at the same stage.

Feedback architectures can be divided into single-path delay feedback (SDF)[1],[14] which process a continuous flow of one sample per clock cycle, and multi-path delay feedback (MDF) or parallel feedback[4], which process several samples in parallel.

On the other hand, feed forward architectures also known as multi-path delay commutator (MDC)[12], do not have feedback loops and each stage passes the processed data to the next stage. These architectures can also process several samples in parallel. In current real-time applications, the FFT has to be calculated at very high throughput rates, even in the range of Giga samples per second. These high-performance requirements appear in applications such as orthogonal frequency division multiplexing (OFDM)[5] and ultra wideband (UWB)[8],[13] .

Two main challenges can be distinguished. The first one is to calculate the FFT of multiple independent data sequences. In this case, all the FFT processors can share the rotation memory in order to reduce the hardware. Designs that manage a variable number of sequences can also be obtained. The second challenge is to calculate the FFT when several samples of the same sequence are received in parallel. This must be done when the required throughput is higher than the clock frequency of the device. In this case it is necessary to resort to FFT architectures that can manage several samples in parallel. However, radix-2^{16} had not been considered for feed forward architectures until the first radix-2^2 feed forward FFT architectures were proposed a few years ago. As a result, parallel feedback architectures [4],[7],[15] which had not been considered for several decades, have become very popular in the last few years. Conversely, not very much attention has been paid to feed forward (MDC) architectures. This paradoxical fact, however, has simple explanation. Originally, SDF and MDC architecture were proposed for radix-2[6] and radix-4[2]. Some years later, radix-2^k was presented for the SDF[1],[12] FFT improvement on radix-2 and radix-4[2]. Next, radix-2^3 and radix-2^3, which enable certain complex multipliers to be simplified, were also presented for the SDF FFT. Finally, the current need for high throughput has been meet by the MDF, which includes multiple interconnected SDF paths in parallel.

The proposed architecture presents the Pipelined radix-2^{16} feedforward FFT architectures. The proposed MDF architecture can provide a higher throughput rate with minimal hardware cost by combining the features of MDC and SDF. The MDF architecture has lower hardware cost compared with the traditional SDF approach and adopts the radix-2^{16} FFT architecture to reduce power dissipation.

2. Fast Fourier Transform

The Fast Fourier Transform algorithm exploit the two basic properties of the twiddle factor - the symmetry property and periodicity property which reduces the number of complex multiplications required to perform DFT. FFT algorithms are based on the fundamental principle of decomposing the computation of discrete Fourier Transform of a sequence of length N into successively smaller discrete Fourier transforms. There are basically two classes of FFT algorithms. Decimation in Time (DIT) algorithm and Decimation in Frequency (DIF) algorithm. In decimation-in-time, the sequence for which we need the DFT is successively divided into smaller sequences and the DFTs of these subsequences are combined in a certain pattern to obtain the required DFT of the entire sequence. In the decimation-in-frequency approach, the frequency samples of the DFT are decomposed into smaller and smaller subsequences in a similar manner. The number of complex multiplication and addition operations required by the simple forms both the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) is of order N^2 as there are N data points to calculate, each of which requires N complex arithmetic operations. The discrete Fourier transform is defined by the (1)

$$X(K)=\sum_{n=0}^{N-1} x(n)\bullet e^{\frac{-j2\Pi nK}{N}}; \quad (1)$$

Where K is an integer ranging from 0 to $N-1$.The algorithmic complexity of DFT will O(N^2) and hence is not a very efficient method. If we can't do any better than this then the DFT will not be very useful for the majority of practical DSP application. However, there are a number of different 'Fast Fourier Transform' (FFT) algorithms that enable the calculation the Fourier transform of a signal much faster than a DFT. As the name suggests, FFTs are algorithms for quick calculation of discrete Fourier transform of a data vector. The FFT is a DFT algorithm which reduces the number of computations needed for N points from O(N^2) to O($N \log_2 N$) where log is the base-2 logarithm. If the function to be transformed is not harmonically related to the sampling frequency, the response of an FFT looks like a 'sinc' function (sin x) / x.

3. Radix-2^2 FFT Algorithm

The DFT of an input sequence is defined in (2)

$$X(K)=\sum_{n=0}^{N-1} x(n)\bullet e^{\frac{-j2\Pi nK}{N}}; \quad k=0,1\ldots.N-1 \quad (2)$$

When N When is a power of two, the FFT based on Cooley-Tukey algorithm is most commonly used in order to compute the DFT efficiently. The Cooley-Tukey algorithm reduces the number of operations from O(N^2) for the DFT to O($N \log_2 N$) for the FFT. In accordance with this, the FFT is calculated in a series $n=\log_\rho N$ of stages, where ρ is the base of the radix, r, of the FFT, i.e., $r = \rho^\alpha$. Flow graphs of 16-point radix-2 and radix-2^2 using decimation in frequency(DIF).The Comparison of Execution Times, DFT & Radix – 2 FFT is tabulated in Table 1.At each stage of the graphs, S€{1,…..,n}, butterflies and rotations have to be calculated. The lower edges of the butterflies are always multiplied by -1.These -1 are not depicted on order to simplify the graphs. Flow graph of 16-point radix-2 represent in the Figure 1.The numbers at the input represent the index of the input sequence, whereas those at the output are the frequencies, k, of the output signal $X[k]$ .Finally each number, Φ, in between the stages indicates a rotation by (3) As a consequence, samples for which $\Phi=0$ do not need to rotated likewise, if Φ € [0,N/4,N/2,3N/4] the samples must be rotated by 0^o,270^o,180^o and 90^o which correspond to complex multiplication by 1,-j,-1,j respectively. These rotations are considered trivial, because they can be performed by interchanging the real and imaginary components and/or changing the sign of data.

$$W_N^{\Phi} = e^{-j\frac{2\pi}{N}\Phi} \quad (3)$$

Table 1: Comparison of Execution Times, DFT & Radix- 2

Number of Points, N	*Complex Multiplications in Direct computations, N^2*	*Complex Multiplication in FFT Algorithm, (N/2) $\log_2 N$*	*Speed improvement Factor*
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1024	80	12.8
64	4096	192	21.3
128	16384	448	36.6

Radix-2^2 is based on radix -2 and the flow graph of a radix-2^2 DIF FFT can be obtained from the graph of a radix-2 DIF one. This This can be done by breaking down each angle Φ, at odd stages into a trivial rotation and a non-trivial one, Φ', where $\Phi'= \Phi$ mod N/4, and moving the latter to the following stage. This is possible thanks to the fact that in the radix-2 DIF FFT the rotation angles at the two inputs of every butterfly, Φ_A and Φ_B , only differ by 0 or N/4. Thus, if $\Phi_A= \Phi'$ and $\Phi_B= \Phi'$+N/4 , the rotation is moved to the following stage. Where the first side of (4) represents the computations using radix-2 and the second one using radix-2^2, and being the input data of the butterfly.

In radix-2, A and B are rotated before the butterfly is computed, whereas in radix-2^2 is rotated by the trivial rotation –j before the butterfly, and the remaining rotation is carried out after the butterfly. Consequently, rotations by Φ' can be combined with those rotations of the following stage.

$$A\cdot W_N^{\Phi'} \pm B\cdot W_N^{\Phi'}\cdot W_N^{N/4} = [A \pm (-j)B]\cdot W_N^{\Phi'} \quad (4)$$

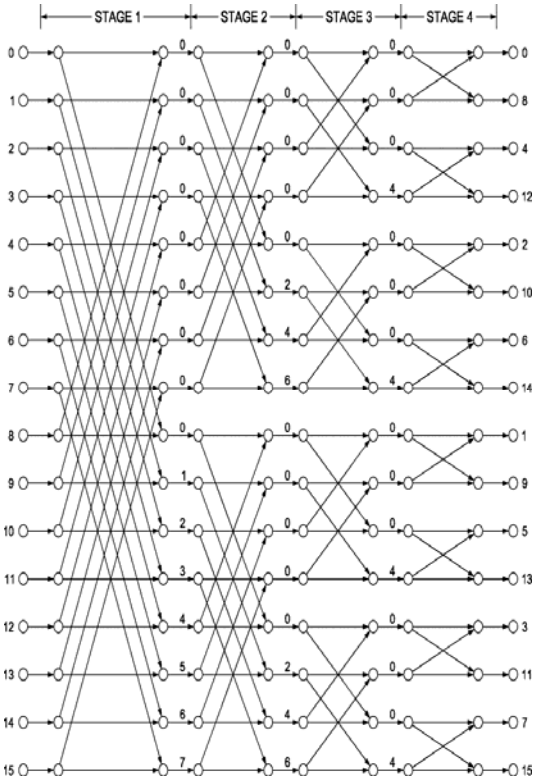

Figure 1: Flow graph of the 16-point radix-2 DIF FFT

4. Radix-2^2 FFT Architectures

The proposed is based on analyzing the flow graph of the FFT and extracting the properties of the algorithm. These properties are requirements that any hardware architecture that calculates the algorithm must fulfill. The properties of the radix-2^2 FFT are shown in Table 2. The following paragraphs explain these properties and how they are obtained.The properties depend on the index of the data, $I \equiv b_{n-1},\ldots\ldots, b_1, b_0$,where ($\equiv$) will be using throughput the paper to relate both decimal and the binary representations of a numbers.

On the one hand, the properties related to the butterfly indicate which samples must be operated together in the butterflies.This condition b_{n-s} is both for DIF and DIT decompositions and means that at any stage of the FFT,*s*, butterflies operate in pairs of data whose indices differ only in bit b_{n-s} , where $n= \log_2 N$ is the number of stages of the FFT. In Figure 2 it can be observed that at the third stage,*s*=3, data with indices I=12$\equiv$1100 and I'=14$\equiv$1110 are processed together by a butterfly. These indices differ in bit b_1, b_{n-s} which meets ,since $n= \log_2 N= \log_2 16=4$ and, thus, $b_{n-s=}\ b_{4-3} = b_1$. On the other hand, there are two properties for rotations. At odd stages of the radix-2^2 DIF FFT only those samples whose index fulfills b_{n-s}. $b_{n-s-1}=1$ have to be rotated. These rotations are trivial and the symbol (.) indicates the logic AND function.

Table 2: Properties of the Radix-2^2 FFT algorithm for DIF and DIT

Properties of Radix-2^2	*DIF*	*DIT*
Butterflies	b_{n-s}	b_{n-s}
Trival Rotators(Odd s)	b_{n-s}. $b_{n-s-1}=1$	$b_{n-s}.b_{n-s-1}=1$
Non-Trival Rotators (Even s)	$b_{n-s+1}+ b_{n-s}=1$	$b_{n-s-1}+ b_{n-s-2}=1$

For the 16-point radix-2^2 FFT in Figure 2 only samples with indices 12, 13, 14, and 15 must be rotated at the first stage. For these indices $b_3.b_2=1$ is fulfilled, meeting the property $b_{n-s}.b_{n-s-1}=1$, since n=4 and $s=1$. Conversely, at even stages rotations are non-trivial and they are calculated over indexed data for which $b_{n-s+1}+ b_{n-s}=1$, where the symbol (+) indicates the logic OR function.

5. Radix Feedforward FFT Architecture

This section presents the radix-2^2 feedforward architectures [3]. First, a 16-point and 4-parallel radix-2^2 feedforward FFT architecture is explained in depth in order to clarify the approach and Show how to analyze the architectures.Then, radix-2^2 feedforward [11] architectures for different number of parallel samples are presented. Figure 2 represent the 4-parallel radix-8 feedforward [11] FFT architecture. The architecture is made up of radix-2[6] butterflies (R2), non-trivial rotators, trivial rotators, which are diamond- shaped, and shuffling structures, which consist of buffers and multiplexers. The lengths of the buffers are indicated by a number. The architecture processes four samples in parallel in a continuous flow. The order of the data at the different stages is shown at the bottom of the figure 2 by their indices, together with the bits b_i that correspond to these indices. In the horizontal, indexed samples arrive at the same terminal at different time instants, whereas samples in the vertical arrive at the same time at different terminals. Finally, samples flow from left to right. Thus, indexed samples (0, 8, 4,12) arrive in parallel at the inputs of the circuit at the first clock cycle, whereas indexed samples (12, 13, 14, 15) arrive at consecutive clock cycles at the lower input terminal. Taking the previous considerations into account, the architecture can be analyzed as follows. First, it can be observed that butterflies always operate in pairs of samples whose indices differ in bit b_{n-s}, meeting the property in Table 2. For instance, the pairs of data that arrive at the upper butterfly of the first stage are: (0, 8), (1, 9), (2, 10), and (3, 11).The binary representation of these pairs of numbers only differs in b_3. As $n=4$, and $s=1$ at the first stage,$b_{n-8}=b_{4-1}=b_3$, so the condition is fulfilled. This property can also be checked for the rest of the butterflies in a similar way that rotations at odd stages are trivial and only affect samples whose indices fulfill b_{n-s} .$b_{n-s-1}=1$.By particularizing this condition for the first stage, b_3. $b_2=1$ is obtained. In the architecture shown in Figure 2 the indices that fulfill this condition are those of the lower edge and, thus, a trivial rotator is included at that edge. On the other hand, the condition for non-trivial rotations at even stages is $b_{n-s+1}+b_{n-s}=1$,$b_3+b_2=1$, being for the second stage. As $b_3+b_2=0$ for all indexed samples at the upper edge of the second stage, this edge does not need any rotator. Conversely, for the rest of edges $b_3+b_2=1$, so they include non-trivial rotators.

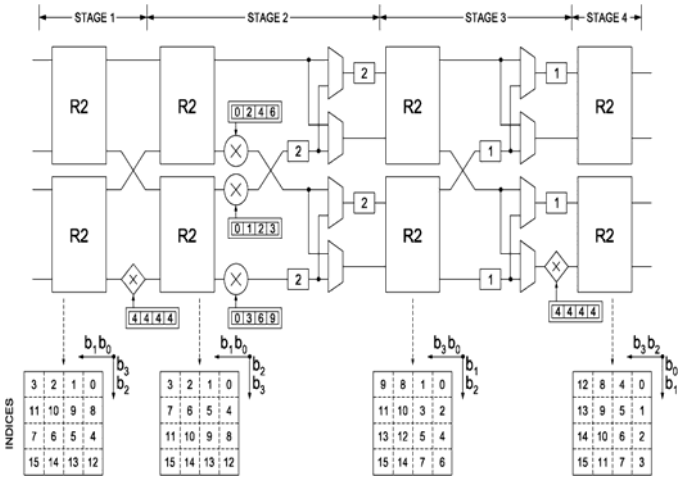

Figure 2: Proposed 4-prellel Radix-2 feedforward architecture for the computation of the 16-point DIF FFT

The rotation memories of the circuit store the coefficients Φ of the flow graph. It can be seen that the coefficient associated to each index is the same as that in the flow graph of Figure 1.For instance, at the flow graph the sample with index $I = 14$ has to be rotated by at the second stage. In the architecture shown in Figure 3 the sample with index is the third one that arrives at the lower edge of the second stage. Thus, the third position of the rotation memory of the lower rotator stores the coefficient for the angle $\Phi = 6$.

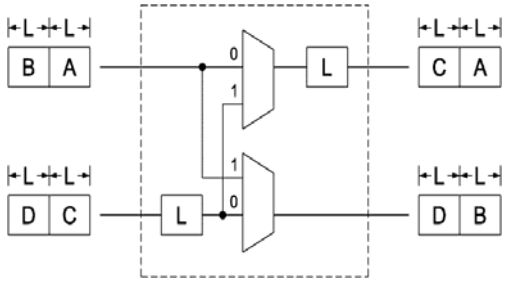

Figure 3: Circuit for data shuffling

Thirdly, the buffers and multiplexers carry out data shuffling. These circuits have already been used in previous pipelined FFT architectures, and Figure 2 shows how they work. For the first L clock cycles the multiplexers are set to "0" L, being the length of the buffers. Thus, the first samples from the upper path (set A) are stored in the output buffer and the first samples from the lower path (set C) are stored in the input buffer. Next, the multiplexer changes to "1", so set C passes to the output buffer and set D is stored in the input buffer. At the same time, sets and are provided in parallel at the output. When the multiplexer commutes again to "0", sets C and D are provided in parallel. As a result, sets B and C are interchanged.Finally, the control of the circuit is very simple: As the multiplexers commute every L clock cycles and L is a power of two, the control signals of the multiplexers are directly obtained from the bits of a counter, in the proposed architectures the number of butterflies depends on to the number of samples in parallel, $P = 2^p$. For any P parallel N -point FFT the number of butterflies is $P/2$ $log2\ N=P\ log4\ N$. Therefore, the number of complex adders is $2P\ log_4 N$. Likewise, the number of rotators is $3P/4\ (log_4 N-1)$. The only exception is for $P=2$. In this case, the number of rotators is $2\ (log_4 N-1)$ .The proposed architectures can process a continuous flow of data. The throughput in samples per clock cycle number of samples in parallel $P=2^p$, whereas the latency is proportional to the size of the FFT divided by the number of parallel samples, i.e. N/P, . Thus, the most suitable architecture for a given application can be selected by considering the throughput and latency that the application demands. Indeed, the number of parallel samples can be increased arbitrarily, which assures that the most demanding requirements are met. Finally, the memory size does not increase with the number of parallel samples. For the architectures shown in, the shuffling structure at any stage $s \in [p,n-1]$ requires $P=2^p$ buffers of length $L=N/2^{s+1}$. According to this, the total sample memory of the architectures is represented in (5)

$$\sum_{S=P}^{N-1} 2^P.L = \sum_{S=P}^{Log_2 N-1} 2^p \frac{N}{2^{s+1}} = N-2^P = N-P \qquad (5)$$

Therefore, a total sample memory of N addresses is enough for the computation of an N-point FFT independently of the degree of parallelism of the FFT. Indeed, the total memory of N-P addresses that the proposed architectures require is the minimum amount of memory for an N-point P-parallel FFT.

6. Experimental Results

The presented architectures have been programmed for the use in field-programmable gate arrays (FPGAs).The designs are parameterizable in the number of points, word length, and number of samples in parallel. Table 3 shows post-place and route results for different configurations of N and P=4, using a word length of 16 bits. The target FPGA is a Virtex-5 FPGA, XC3S500E. In the proposed designs these blocks have been used to implement complex multipliers that carry out the rotation of the FFT. Figure 4 compares the area of the proposed architectures to other equivalent high-throughput pipelined FFTs architectures for the same FPGA and synthesis conditions. Full streaming architectures (FS) have been generated using the tool presented, which provides optimized pipelined architectures for a given radix and number of parallel samples. The results for 4-parallel pipelined architectures are shown in Figure 2 In the figure 4, the numbers next to the lines indicate the amount slices that each architecture requires. It can be observed that the proposed radix- architectures require less area than previous designs for any FFT size, this improvement increases with the size of the FFT.

Table 3: Area and Performance Of The Proposed 4-Parallel N Point Radix-2 Feedforward FFT Architectures

FFT *P=4* *N*	Area Slices	Latency (ns)	Freq. (MHz)	Throughput (MS/s)
16	386	26	458	1831
64	695	81	389	1554
256	1024	221	384	1536
1024	1425	1055	270	1081
4096	2388	6120	173	693

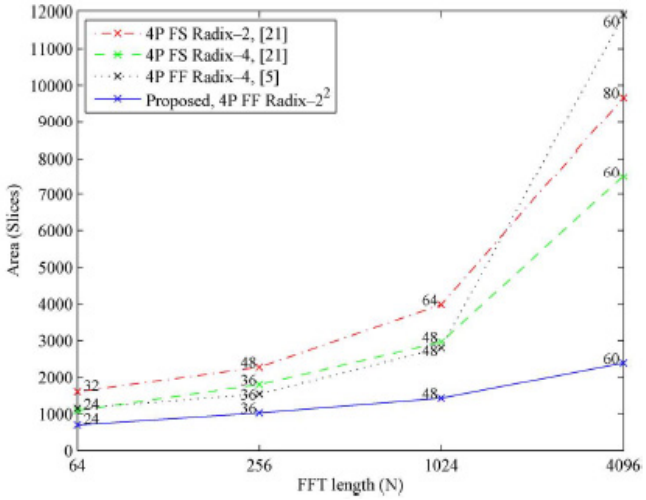

Figure 4: Area of 4-parallel pipelined FFT Architecture

7. Conclusion

This study extends the use of radix-2 to feedforward (MDC) FFT architectures. Indeed, it is shown that feedforward structures are more efficient than feedback ones when several samples in parallel must be processed. In feedforward architectures radix-2 can be used for any number of parallel samples which is a power of two. Indeed, the number of parallel samples can be chosen arbitrarily depending of the throughput that is required. Additionally, both DIF and DIT decompositions can be used. Finally, experimental results show that the designs are efficient both in area and performance, being possible to obtain throughputs of the order of GigaSamples/s as well as very low latencies.

References

[1] Cortés, I. Vélez, and J. F. Sevillano, "Radix FFTs: Matricial representation and SDC/SDF pipeline implementation," IEEE Trans.Signal Process., vol. 57, no. 7, pp. 2824–2839, Jul. 2009.
[2] A. M. Despain, "Fourier transform computers using CORDIC iterations," IEEE Trans. Comput., vol. C-23, pp. 993–1001, Oct. 1974.
[3] C. Cheng and K.K. Parhi, "High-throughputVLSI architecture for FFT computation," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 10,pp. 863–867, Oct. 2007.
[4] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," IEEE Trans. Comput., vol.C-33, no. 5, pp. 414–426, May 1984.
[5] H. Liu and H. Lee, "A high performance four-parallel 128/64-point radix- FFT/IFFT processor for MIMO-OFDM systems," in Proc.IEEE Asia Pacific Conf. Circuits Syst., 2008, pp. 834–837.
[6] H. L. Groginsky and G. A.Works, "A pipeline fast Fourier transform," IEEE Trans. Comput., vol. C-19, no. 11, pp. 1015–1019, Oct. 1970.
[7] J. Lee, H. Lee, S. I. Cho, and S.-S. Choi, "A high-speed, low-complexity radix- FFT processor for MB-OFDM UWB systems," in Proc. IEEE Int. Symp. Circuits Syst., 2006, pp. 210–213.
[8] L. Liu, J. Ren, X. Wang, and F. Ye, "Design of low-power, 1 GS/s throughput FFT processor for MIMO-OFDM UWB communication system," in Proc. IEEE Int. Symp. Circuits Syst., 2007, pp. 2594–2597.
[9] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 53, no. 7, pp. 585–589, Jul. 2006.
[10] M. A. Sánchez, M. Garrido,M. L. López, and J. Grajal, "Implementing FFT-based digital channelized receivers on FPGA platforms," IEEE Trans. Aerosp. Electron. Syst., vol. 44, no. 4, pp. 1567–1585, Oct. 2008.
[11] N. Li and N. P. van der Meijs, "A radix based parallel pipeline FFT processor for MB-OFDM UWB system," in Proc. IEEE Int. SOC Conf., 2009, pp. 383–386.
[12] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in Proc. IEEE Custom Integr. Circuits Conf., 1998, pp. 131–134.
[13] S.-I. Cho, K.-M. Kang, and S.-S. Choi, "Implemention of 128-point fast Fourier transform processor for UWB systems," in Proc. Int.Wirel. Commun. Mobile Comput. Conf., 2008, pp. 210–213.
[14] W. Xudong and L. Yu, "Special-purpose computer for 64-point FFT based on FPGA," in Proc. Int. Conf. Wirel. Commun. Signal Process. 2009, pp. 1–3.
[15] S.-N. Tang, J.-W. Tsai and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," IEEE Trans. Circuits Syst. I,Reg. Papers, vol. 57, no. 6, pp. 451–455, Jun. 2010.

Author Profile

A. Salai Kishwar Jahan received the Under Graduate Electronics & Communication Engineering from Mount Zion College of Engineering & Technology, India at 2012.Pursuing M.E, VLSI Design in Mother Terasa College Of Engineering Technology, India at 2014. Her field of interest includes VLSI and Digital Signal Process.

A. Indhumathi received the Under Graduate in Electronics & Communication Engineering from Shanmuganathan Engineering College, India at 2006. Post Graduate Communication System from Sudharsan Engineering College, India at 2010. Her field of interest includes VLSI and Digital Signal Process.