National Conference on Innovative Research & Industrial Applications

www.worldconferences.org

A Generalized Algorithm and Efficient Architecture for Mixed-Decimation MDF Structure for Radix-2^k Parallel FFT

SK. KARISHMA[1], K. DHANUNJAYA[2]

[1]PG Scholar, Dept of ECE, Audisankara College of Engineering and Technology (Autonomous), Gudur, AP, India.
[2]HOD, Dept of ECE, Audisankara College of Engineering and Technology (Autonomous), Gudur, AP, India.

Abstract: This paper presents a mixed-decimation multipath delay feedback (M2 DF) approach for the radix-2*k* fast Fouriertransform. We employ the principle of folding transformat ionto derive the proposed architecture, which activates the idle period of arithmetic modules in multipath delay feedback (MDF)architectures by integrating the decimation-in-time operations into the decimation-in-frequency-operated computing units. Furthermore, we compare the proposed design with other efficient schemes, namely, the MDF and the multipath delay commutator (MDC) scheme theoretically and experimentally. Relyingon the obtained expressions and statistics, it can be concluded that the M2DF design serves as an efficient alternative to the MDF scheme, since it achieves improved efficiency in the utilization of arithmetic resources without deteriorating the superiorities of feedback structures. In addition, the recommended design performs better in memory requirement and computing delay compared with the MDC approach.

Keywords: Decimation-In-Frequency (DIF), Decimation-In time (DIT), Fast Fourier Transform (FFT), Multipath Delay feedback (MDF), Pipelined-Parallel Architecture.

I. INTRODUCTION

Being an efficient algorithm for discrete Fourier transform (DFT) computation, fast Fourier transform (FFT)has seen broader usage in the field of digital signal processing.It also plays an increasing important role in modern digital communications, since orthogonal frequency division multiplexing technique has been adopted in leading communication standards, including wireless LAN and long-term evolution. Therefore, an efficient implementation of FFT has attracted much attention and hardware designers have put forward various schemes to achieve reasonable tradeoffs between area and performance. By shortening the critical path in the signal diagram, pipelined architectures have an inherent advantage over other efficient hardware structures in providing high throughputs. To cope with the gradual increase of real-timebusiness, plenty of published works have focused on designingand optimizing the pipelined structures [1]–[21]. In the serial-input-serial-output (SISO) scenario, single-path delay commutator (SDC) [1] structure is one of the mostclassical approaches to perform the pipelined FFTcomputation. To reduce the memory banks in SDC pipelines,single-path delay feedback (SDF) architecture is proposedin [2], which is characterized by the feedback connectionsin the circuits. These hardware schemes can be combinedwith the radix-2 [2], [3], radix-4 [4], and especially radix-2*k*algorithm [5]–[8] to execute the DFT operation. Comparedwith the radix-4 approach, the radix-2*k* pipeline is equippedwith simpler butterfly units while making a better utilizationof complex multipliers than the typical radix-2 scheme.

Thus,radix-2*k* algorithm acts as an effective alternative to theconventional computation methods from the perspective ofhardware design. Admittedly, the extension of communication service hasstimulated a dramatic rise of throughput requirements. Thisadvance in real-time business has additionally rendered thehardware schemes designed for SISO applications obsoleteto a certain extent. On this occasion, multipath delay commutator (MDC) [9]–[13] and multipath delayfeedback (MDF) [14]–[21], which serve as the upgrade of SDCand SDF, respectively, are proposed to calculate the FFT whenseveral samples of the same sequence are received in parallel.In general, the MDF structure is composed of multipleinterconnected SDF paths, and each path is responsiblefor managing one of the parallel input streams. This designcontributes to the inheritance of utilizing registers efficiently atthe expense of squandering the arithmetic components, particularlythe butterfly units. By contrast, the MDC approach pavesthe way for boosting the hardware efficiency of arithmeticunits (AUs), while additional memories have to be consumedfor either reordering the samples or folding the streams,which additionally leads to an increase of computing delay. Whether feed forward structures represented by the MDCapproach or feedback structures, such as the SDF andthe MDF design, they afford feasible solutions to strikea balance between the consumption of hardware resourcesand the reachable performance.

As we move forward the discussion, the hardware resources can be further divided intotwo categories: 1) arithmetic resources associated with logicalor arithmetic operations and 2) memory resources referred toas memory units, which are responsible for caching samples.Due to the outstanding performance in the efficient use ofmemory resources, the

MDF scheme has been a tremendoussuccess in a variety of applications [17]–[21]. Underneaththe triumph, the underutilization of arithmetic resources isstill a stubborn problem for feedback design and has notbeen resolved satisfactorily. The objective of this paper is to make a breakthrough in this respect while maintaining theadvantages of feedback structures. To this end, the theory offolding transformation described in [22] and first adopted toderive the pipelined FFT architectures in [11] is employedto derive the proposed scheme, namely, the mixed-decimation MDF (M2DF) architecture. The kernel of M2DF design lies inscheduling the decimation-in-time (DIT) operations onto thedecimation-in-frequency (DIF)-operated basic blocks. By thismeans, we mobilize the idle period of arithmetic modules inMDF architectures and thereby gain a considerable reductionof arithmetic resources to make up for the deficiency of thefeedback FFT modules. The rest of this paper is organized as follows. In Section II,we introduce the top-level architecture of the pipelined-parallelFFT processor based on the matrix factorization ofradix-2*k* algorithm. Section III expands on the M2DFarchitectures using folding transformation. Theoretical andexperimental comparisons of the M2DF scheme and existingapproaches are related in Section IV. Finally, the conclusionis drawn in Section V.

II. DESIGN OF MIXED-DECIMATION MDFARCHITECTURE

In this section, we first apply the theory of folding transformationto derive the folding matrices associated with DIF- andDIT-operated SDF processor. With the assist of this theoretical basis, the operations in SDF pipeline are rescheduled toreverse the underutilization of arithmetic modules,which isaccomplished by integrating DIT operations into DIF-operatedbasic blocks. Finally, this optimization strategy is used toimplement the top-level design in Section II, generating theproposed M2DF structure with improved hardware efficiency.

A.Analyzing the SDF Scheme using Folding Transformation

The folding transformation provides a systematic techniquefor designing control circuits for hardware where severalalgorithm operations are time multiplexed on a single computingdevice [22]. Consider an eight-point radix-2*k* DIFcomputation, the algorithm can be presented using a data flowgraph shown in Fig. 2(a), where the nodes represent computationsand the directed edges represent data paths. As shown,the flow graph consists of three stages and four operationsare executed within each stage. When *xi* , *i* = 0, . . . , 7arrives in serial, multiple operations in each stage can betime multiplexed to a single computing unit without anycollisions, and the control circuits are determined systematicallyby folding transformation. In this way, the FFTflow graph in Fig. 2(a) can be transformed into a pipelinedform shown in Fig. 2(b), where operations A_0, . . . , A_3,B_0, . . . , $B3$, and C_0, . . . ,C_3 are performed in the computingunit U_A, U_B, and U_C, respectively. The multiple operations included in a computing moduleare arranged by folding sets. A folding set is an ordered setof operations executed by the same computing unit. Apartfrom the operations associated with the nodes in the dataflow graph, generally, there are also null operations in thefolding set. The number of operations folded into a foldingset is called the folding factor (denoted by *R* in subsequentdiscussion). Correspondingly, the computing unit works incycles with a period taking up *R* time partitions, and theoperation in the *r* th (*r* = 0, . . . , *R* − 1) position is executed

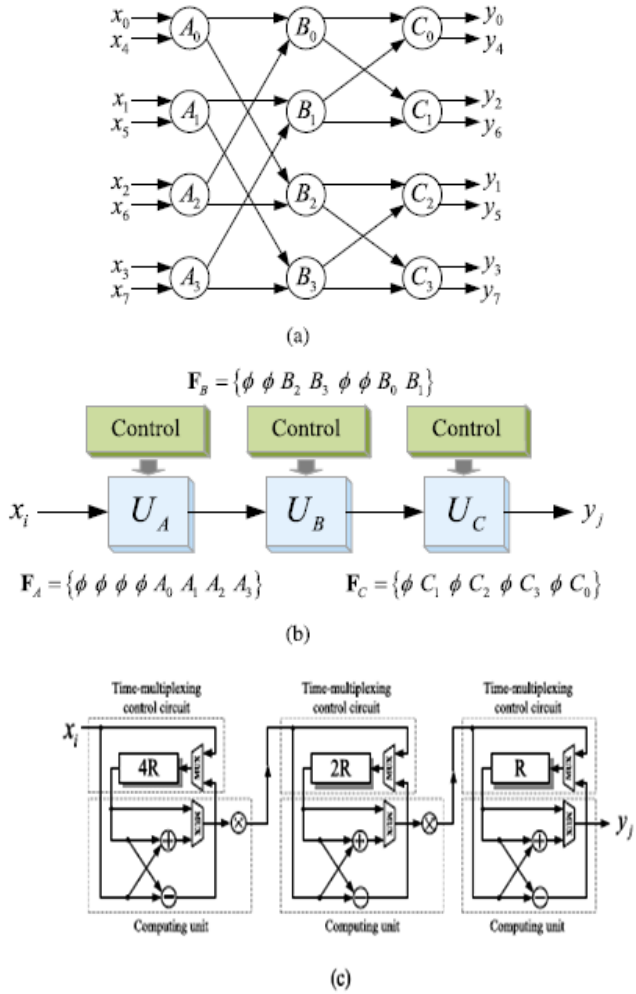

Fig.1. (a) Data flow graph of an eight-point DIF DFT using a radix-2k algorithm. (b) Data flow graph can be converted into a pipelined version through folding transformation. (c) Optimized hardware scheme to implement the pipelined flow graph. Moreover, the shift register unit marked n R indicates n positions are available in the device to cache the complex inputs.

During the *r*th time partition. For example, consider a foldingset **F** = {*φ A*0 *φ A*1 *φ A*2 *φ A*3} with *R* = 8, the computingunit operates *A*0, *A*1, *A*2, and *A*3 in the first, third, fifth,and seventh time partitions, respectively, while keeping idlein remaining intervals.When folding transformation is utilized to derive the circuitfrom a data flow graph, the accurate determination of foldingsets serves as the cornerstone. In particular, the foldingfactor depends on the periodicity of operations executed bya computing unit, while the arrangement of elements withina folding set is linked closely to specific hardware structures.In terms of an *N*-point

SDF processor using a radix-2*k* DIFFFT algorithm, it constitutes of log2*N* computing modules.The *l*th (*l* = 0, 1, . . . ,log2*N* − 1) module alternates idlecondition and operating status with an interval, including*N*/2*l*+1 time instances. Taking this priori knowledge intoaccount, the folding sets corresponding to *UA*, *UB*, and *UC*in Fig. 2(b) can be written preliminarily as

$$\check{\mathbf{F}}_A = \{\phi \;\; \phi \;\; \phi \;\; \phi \;\; A_0 \;\; A_1 \;\; A_2 \;\; A_3\} \tag{1a}$$

$$\check{\mathbf{F}}_B = \{\phi \;\; \phi \;\; B_0 \;\; B_1 \;\; \phi \;\; \phi \;\; B_2 \;\; B_3\} \tag{1b}$$

$$\check{\mathbf{F}}_C = \{\phi \;\; C_0 \;\; \phi \;\; C_1 \;\; \phi \;\; C_2 \;\; \phi \;\; C_3\} \tag{1c}$$

with the folding factor R=8.

Equations(1a)–(1c) incorporate the features of SDF computing units into the derivation of folding sets. Nevertheless, they fail to consider the precedence of operationsassociated with different modules. As can be found inFig. 2(b), when the data stream flows serially to *UA*, the newlygenerated intermediate results would arrive at *UB* and *UC* witha delay of four and six time partitions, respectively, after whichthe modules are able to perform the operations formulatedby (1b) and (1c). In view of this causality constraint, the *m*th (*m* = 0, . . . , *R*−1) element in ˇ**F** *B* ought to be aligned withthe [*m* + 4]*R*th entry of ˇ**F** *A* ([·]*x* returns the modulus of *x*).Likewise, the *n*th (*n* = 0, . . . , *R* − 1) element in ˇ**F***C* requiresto match the [*n* + 6]*R*th position of ˇ**F** *A*. This yields modifiedfolding sets

$$\begin{aligned}\mathbf{F}_A &= \{\phi \;\; \phi \;\; \phi \;\; \phi \;\; A_0 \;\; A_1 \;\; A_2 \;\; A_3\}\\ \mathbf{F}_B &= \{\phi \;\; \phi \;\; B_2 \;\; B_3 \;\; \phi \;\; \phi \;\; B_0 \;\; B_1\}\\ \mathbf{F}_C &= \{\phi \;\; C_1 \;\; \phi \;\; C_2 \;\; \phi \;\; C_3 \;\; \phi \;\; C_0\}\end{aligned} \tag{2}$$

Where **F***A* inherits ˇ**F** *A* directly. By contrast, **F***B*, **F***C* can beobtained by cyclic right shifting the elements in ˇ**F** *B*, ˇ**F***C* forfour and six positions, respectively. As folding sets reveallimitations when analyzing the entire hardware structure, werecommend an alternative analytical tool named folding matrixto cover this shortage. Denote an *n f* × *R* matrix

$$\mathbf{F}_{\mathrm{DIF}} = \begin{bmatrix} \phi & \phi & \phi & \phi & A_0 & A_1 & A_2 & A_3 \\ \phi & \phi & B_2 & B_3 & \phi & \phi & B_0 & B_1 \\ \phi & C_1 & \phi & C_2 & \phi & C_3 & \phi & C_0 \end{bmatrix} \tag{3}$$

as the folding matrix corresponding to (1a)–(1c), where*n f* equals the amount of computing stages in the folded flowgraph, while the rows of **F**DIF corresponds to different foldingsets. Therefore, the entry *(***F**DIF*)m,n*(*m* = 0, . . . , *n f* − 1,*n* = 0, . . . , *R*−1) represents the operation executed by the *m*th computing unit at the *l R* + *n*th (*l* = 1, 2, . . .) time instance. When obtaining the folding sets associated with *UA*, *UB*, and *UC*, the supporting circuits used to fulfillthe time multiplexing of computing units can be obtainedthrough folding transformation. To boost the hardwareefficiency, the implementation scheme can be furtheroptimized by introducing the register minimization technique,which eventually generates the SDF hardware structurepresented in Fig. 2(c). The shift register unit marked *n*Rin the graph indicates that *n* positions are available in thecomponent to cache the complex inputs. Similar processes can be applied to analyze the DIT dataflow graph shown in Fig. 3(a), which is essentially a transpositionof its DIF counterpart. Assuming that the input samplesarrive in bit-reversed order, the folding sets considering theproperty of SDF computing are

$$\begin{aligned}\check{\mathbf{G}}_{\tilde{A}} &= \{\phi \;\; \tilde{A}_0 \;\; \phi \;\; \tilde{A}_1 \;\; \phi \;\; \tilde{A}_2 \;\; \phi \;\; \tilde{A}_3\}\\ \check{\mathbf{G}}_{\tilde{B}} &= \{\phi \;\; \phi \;\; \tilde{B}_0 \;\; \tilde{B}_1 \;\; \phi \;\; \phi \;\; \tilde{B}_2 \;\; \tilde{B}_3\}\\ \check{\mathbf{G}}_{\tilde{C}} &= \{\phi \;\; \phi \;\; \phi \;\; \phi \;\; \tilde{C}_0 \;\; \tilde{C}_1 \;\; \tilde{C}_2 \;\; \tilde{C}_3\}\end{aligned} \tag{4}$$

with the folding factor *R* = 8. Note that the arrival of freshlygenerated results at *U* ˜*B* and *U* ˜*C* involve additional delay ofone and three time partitions, thus by cyclic right shifting the

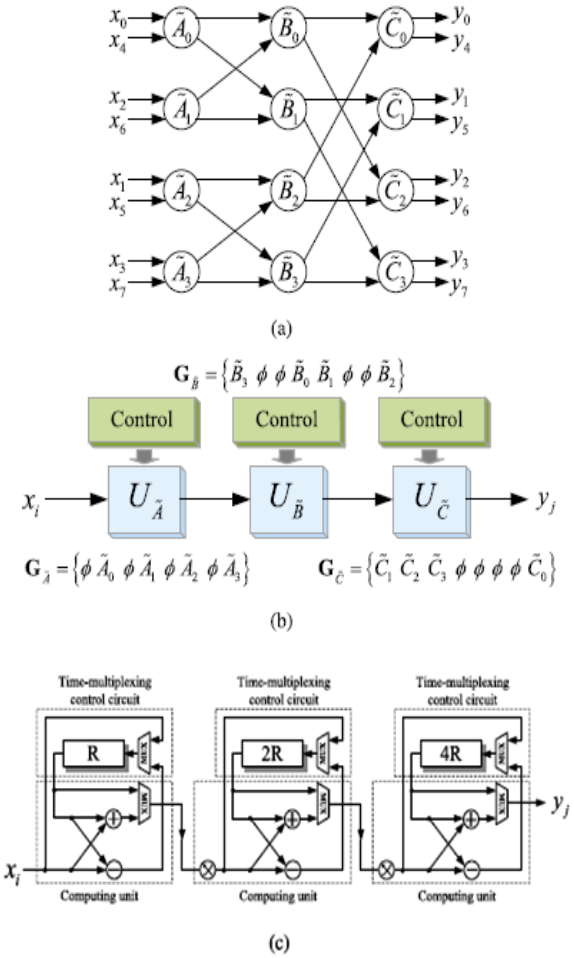

Fig.2. (a) Data flow graph of an eight-point DIT DFT using a radix-2k algorithm. (b) Data flow graph can be converted into a pipelined version through folding transformation. (c) Optimized hardware scheme to implement the pipelined flow graph.Elements in ˇG˜*B*for one space and ˇG˜*C*for three spaces, weobtain.

$$\begin{aligned}\mathbf{G}_{\tilde{A}} &= \{\phi \;\; \tilde{A}_0 \;\; \phi \;\; \tilde{A}_1 \;\; \phi \;\; \tilde{A}_2 \;\; \phi \;\; \tilde{A}_3\}\\ \mathbf{G}_{\tilde{B}} &= \{\tilde{B}_3 \;\; \phi \;\; \phi \;\; \tilde{B}_0 \;\; \tilde{B}_1 \;\; \phi \;\; \phi \;\; \tilde{B}_2\}\\ \mathbf{G}_{\tilde{C}} &= \{\tilde{C}_1 \;\; \tilde{C}_2 \;\; \tilde{C}_3 \;\; \phi \;\; \phi \;\; \phi \;\; \phi \;\; \tilde{C}_0\}.\end{aligned} \tag{5}$$

Accordingly, the $n\,f \times R$ folding matrix corresponding to theDIT data flow graph is expressed as

$$\mathbf{G}_{\mathrm{DIT}} = \begin{bmatrix} \phi & \tilde{A}_0 & \phi & \tilde{A}_1 & \phi & \tilde{A}_2 & \phi & \tilde{A}_3 \\ \tilde{B}_3 & \phi & \phi & \tilde{B}_0 & \tilde{B}_1 & \phi & \phi & \tilde{B}_2 \\ \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & \phi & \phi & \phi & \phi & \tilde{C}_0 \end{bmatrix}. \quad (6)$$

The entry $(\mathbf{G}\mathrm{DIT})m,n(m = 0, \ldots, n\,f - 1, n = 0, \ldots, R - 1)$also represents the operation executed by the mth computing module at the $l\,R + n$th ($l = 1, 2, \ldots$) time instance. Withthe assist of folding transformation and register minimizationtechniques, the hardware scheme can be derived, which isshown in Fig. 3(c).

B. Rescheduling the Operations in SDF Pipelines

The folding matrix given in (3) and (7) conveys the factthat whether designers adopt the DIF approach or DIT schemeto construct the SDF circuit, the existence of null operations in folding sets will always degrade the efficiency of arithmeticcomponents, leading to an approximate 50% utilization ofcomplex adders merely. To address this issue, we reschedulethe operations in SDF pipelines to activate the idle intervals ofcomputing units. As the folding matrix acts as the simplifiedrepresentation of circuit status, the rearrangement of operationscan be achieved equivalently by modifying the foldingmatrices. To be specific, the modification is accomplishedthrough the following three steps. Cyclic right shift the columns of **G**DIT for $SR = 1$ space, it generates

$$(\mathbf{G}_{\mathrm{DIT}})_1 = \begin{bmatrix} \tilde{A}_3 & \phi & \tilde{A}_0 & \phi & \tilde{A}_1 & \phi & \tilde{A}_2 & \phi \\ \tilde{B}_2 & \tilde{B}_3 & \phi & \phi & \tilde{B}_0 & \tilde{B}_1 & \phi & \phi \\ \tilde{C}_0 & \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & \phi & \phi & \phi & \phi \end{bmatrix} \quad (7)$$

where the subscript outside the brackets denotes thecyclic right shift operation. Note the transformationintroduces an additional SR-unit delay to the foldingsets $\mathbf{G}\,\tilde{}A$, $\mathbf{G}\,\tilde{}B$, and $\mathbf{G}\,\tilde{}C$ simultaneously, thus the entry$[(\mathbf{G}\mathrm{DIT})1]m,n(m = 0, \ldots, n\,f - 1, n = 0, \ldots, R - 1)$represents the operation executed by the mth computing module at the $l\,R + n - 1$th ($l = 1, 2, \ldots$) time instance,differing from the definitions of **G**DIT. To fulfill theoperations formulated by (7), a latency unit providinga SR-clock-cycle delay for input samples should beintegrated into the circuit associated with **G**DIT.

Flip $(\mathbf{G}\mathrm{DIT})1$ vertically to obtain

$$(\mathbf{G}_{\mathrm{DIT}})_1^{\mathrm{F}} = \begin{bmatrix} \tilde{C}_0 & \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & \phi & \phi & \phi & \phi \\ \tilde{B}_2 & \tilde{B}_3 & \phi & \phi & \tilde{B}_0 & \tilde{B}_1 & \phi & \phi \\ \tilde{A}_3 & \phi & \tilde{A}_0 & \phi & \tilde{A}_1 & \phi & \tilde{A}_2 & \phi \end{bmatrix} \quad (8)$$

Where the added superscript stands for the verticalflip. As the rows of folding matrix are connected withspecific computing units, the modification of currentstep would redefine this mapping relation, i.e., the mth ($m = 0, \ldots, n\,f - 2$ row of $(\mathbf{G}\mathrm{DIT})F1$ describes thetime-multiplexing scheme of the $n\,f - m - 1$th computingunit, rather than the mth module as the original **G**DIT.

Overlay **F**DIF with $(\mathbf{G}\mathrm{DIT})F1$. Thus, we can obtain

$$\mathbf{F} = \mathbf{F}_{\mathrm{DIF}} + (\mathbf{G}_{\mathrm{DIT}})_1^{\mathrm{F}} = \begin{bmatrix} \tilde{C}_0 & \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & A_0 & A_1 & A_2 & A_3 \\ \tilde{B}_2 & \tilde{B}_3 & B_2 & B_3 & \tilde{B}_0 & \tilde{B}_1 & B_0 & B_1 \\ \tilde{A}_3 & C_1 & \tilde{A}_0 & C_2 & \tilde{A}_1 & C_3 & \tilde{A}_2 & C_0 \end{bmatrix}. \quad (9)$$

Clearly, the superposition eliminates the preexisting nulloperations, which suggests that the synthesis of DIT SDFpipeline and DIF SDF pipeline is able to fully utilize the arithmeticmodules. With respect to the hardware implementation,as shown in Fig. 4, the mth ($m = 0, \ldots, n\,f - 1$) computingunit in the DIF SDF pipeline should be integrated with the$n\,f - m - 1$th one in the DIT counterpart due to the vertical flipoperation in the second step. In addition, the DIT-processingstream ought to experience a SR-clock-cycle delay up front

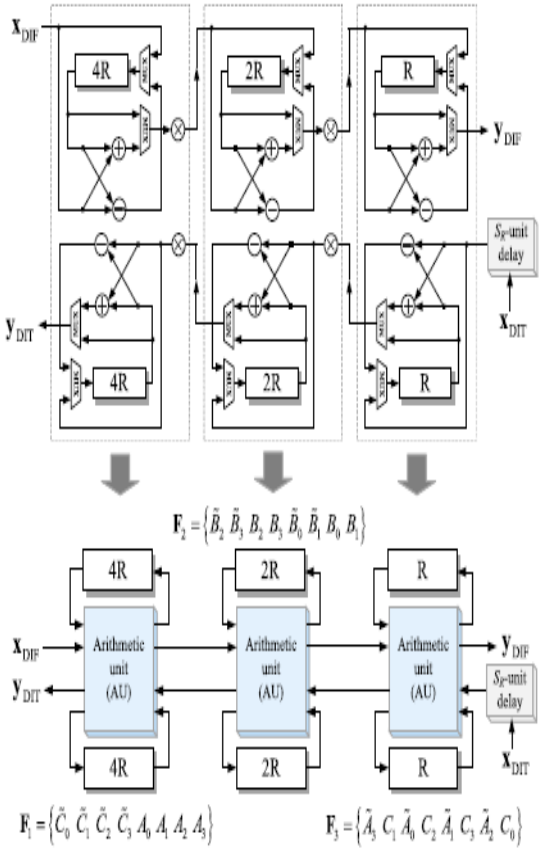

Fig.3. Integration of the DIF and DIT pipelined processor to fully utilize the computing units.

when considering the cyclic shift operation in the first step.It is also worth noting that SR takes non unique value because of the cyclic feature of folding sets. Apart from $SR = 1$ in theprevious discussion, there are a set of candidates

$$S_R = nR + 1, \quad n \in \mathbb{N} \quad (10)$$

with$R$ represents the folding factor, which is equal to theFFT size in serial computing.In terms of the underlying design of arithmetic modules,the conventional SDF elementary

structure should beupgraded to undertake both DIF processing and DIT processingtasks. According to the utilization efficiency of complexmultipliers, two kinds of hardware structures referred to asType I and Type II are proposed in Fig. 5. For Type Iarchitecture in Fig. 5(a), adders are shared by two streams, while multipliers (not exhibited) are provided individually.Type II architecture is shown in Fig. 5(b), where both addersand multipliers are reused to further improve the hardwareefficiency. Thus, Type II is suitable for the situation when datastreams occupy the complex multipliers with a 50% utilizationratio or less. Moreover, the multiplexors and selectors inthe circuit can be divided into two categories (white forCategory I and gray for Category II in Fig. 5). When arithmeticmodules are in service, the components within each categoryshare the identical logic signal. The control scheme, which issynchronized with the DIF-computing stream, is summarizedas follows.

- For the first M samples of the stream, the componentsbelonging to Category I are controlled by logic 1, whileothers in Category II are controlled by logic 0.
- During the next M samples, the control signals shouldbe inverted.

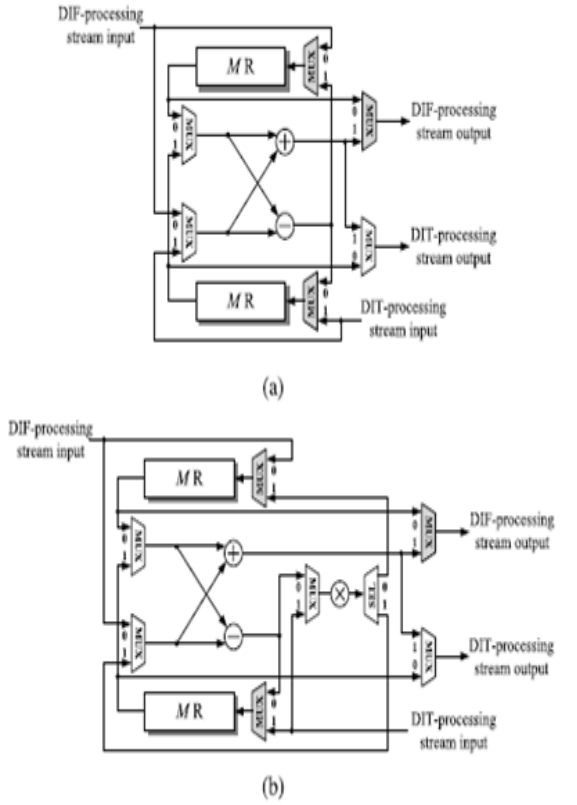

Fig.4 . Underlying structures of AUs. (a) Type I structure, complex adders are shared only. (b) Type II structure, both complex adders and complex multipliers are shared.

The scheme above is suitable for both Type I and Type IIdesign and is 2M-clock-cycle periodic, where M is the lengthof shift register sets in the arithmetic module.

C. Design of the M2DF Hardware Structure

For the top-level design shown in Fig1, the rescheduling strategy proposed in Section III-B is beneficial to optimizethe underlying scheme, which results in the proposedM2DF structure. We begin with two-parallel architecture tounfold the kernel. Taking a 32-point DFT computation, forexample, the corresponding M2DF design is shown in Fig6.During the horizontal DFT processing, as shown, **x**0 and **x**1 Execute DIF and DIT scheme, respectively, to calculate**x**0**T***S* and **x**1**T***S* . By drawing on the discussion in Section III-B,the pipelined circuit constructed by the modified AUs in Fig. 5is able to fulfill this task, which promotes the arithmeticresources to full utilization. In addition, as input samplesrequire to be rearranged as bit reversal before participatingin DIT computation, the latency unit in Fig. 4 is replacedwith a reordering module, which permutes the samples witha delay of $SR = S + 1$ clock cycles. After the rotation, theparallel streams are connected to a simple radix-2 butterflyunit to execute the vertical DFT processing. As for the four-parallel and eight-parallelapplications, Fig. 7 shows the hardware structures usinga 64-point DFT. As shown, the circuit used to performthe horizontal DFT is essentially a parallel extension ofthe counterpart in the two-parallel structure. By contrast, thehardware scheme for vertical DFT depends closely on theparallelism. For four-parallel computing in Fig. 7(a),four radix-2 butterfly units along with requisite rotationdevices operate coordinately to provide a throughput of four samples per clock cycle.

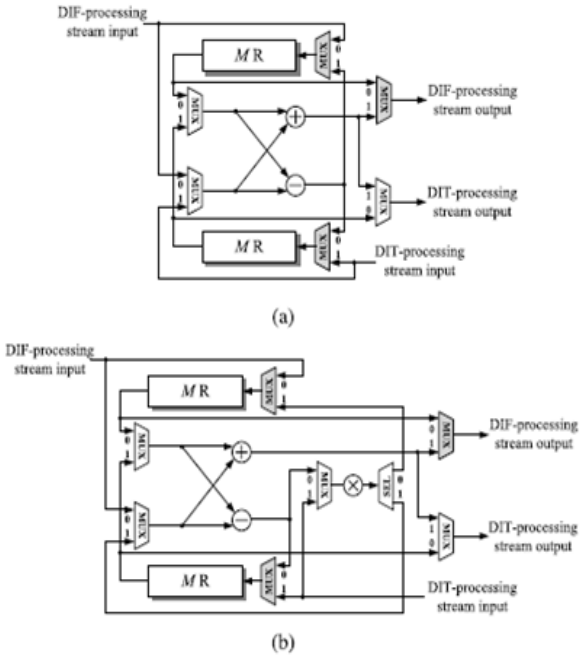

Fig.5. Proposed two-parallel M2DF architectures for the computation of 32-point DFT. (a) Use radix-2 algorithm in the horizontal DFT processing phase. (b) Use radix-22 algorithm in the horizontal DFT processing phase.

In the eight-parallel scenario, by means of rearranging the interconnections of theeight-point radix-2 data flow graph, the multiple stagesincluded in the vertical DFT circuit can be

implementedusing a fixed structure, which contributes to reduce thehardware complexity. Furthermore, note each multiplier in thevertical DFT unit is related to a single known coefficient, theycan be mapped to the low-complexity constant multipliers.The M2DF structure can be readily applied to the *P*-parallelradix-2*k* computation, where even *P* is emphasized here.The exponent *k* will only affect the arrangement of complexmultipliers in the horizontal DFT circuit. On the other hand,the parallelism *P* plays an important role in the realizationof vertical DFT unit, as this module is responsible fora *P*-parallel *P*-point DFT. In terms of the rotation, *P* –1complex multipliers are sufficient to complete the relevanttask, which are irrelevant to the selection of radix-2*k* Algorithm.

III. COMPARISON AND EVALUATION

In this section, we first evaluate the hardware expense of M2DF structures. Afterward, we compare the M2DF scheme with existing approaches theoretically and experimentally.

A. Evaluation of the Proposed M2DF Structure

In general, an FFT processor can be decomposed into three parts: 1) the arithmetic component; 2) the control circuit;and 3) the reordering module. The arithmetic componentis primarily composed of complex adders and multipliersto execute the butterfly operations and twiddle factormultiplications. Moreover, the multipliers can be further categorizedinto the general ones with varied twiddle factorsand constant ones using fixed coefficients. This fine-grainedpartition takes account of the diverse hardware complexitybetween general complex multipliers and constant ones.In this way, the numbers of complex adders λ_a, general complex multipliers $\lambda^g c$and constant complex multipliers accountfor the primary consumption of arithmetic resources in anFFT module. Memory expenditure comes from the realization of controlcircuit and reordering module. The control circuit isresponsible for the rearrangement of data streams to coordinatewith the arithmetic computations. In pipelined structures, shiftregister sets are widely used to implement the control circuit.On the other hand, the reordering module permutes thesamples to obtain the desired parallel output unvec$P,S(\mathbf{y})$. To this end, **Y** in (7b) should be gained firstly, after whichthe permutation network corresponding to the matrix $\mathbf{P}(P)N$is applied to bridge the gap between **Y** and unvec$P,S(\mathbf{y})$.It is worth noting that the hardware design of $\mathbf{P}(P)N$ could beidentical for the FFT processors with the same parallelism,while the circuit to obtain **Y** is linked closely to concreteFFT schemes.

Thus, in the following analysis, the memorycost to complete the first-phase reordering is considered asthe metric to evaluate the hardware efficiency. Furthermore, some FFT processors do not utilize unvec$P,S(\mathbf{x})$ directly, e.g.,Garrido*et al.* [12] convert unvec$P,S(\mathbf{x})$ into [unvec$S,P(\mathbf{x})$]T to launch the FFT. This additional memory requirement is alsoconsidered here. For the M2DF design, $P/2 \cdot \log 2N$ radix-2 butterfly units, which consist of $\lambda a = P \cdot \log 2N$ complex adders are an integralpart of the *P*-parallel, *N*-point DFT computation. The general Complex multipliers are existed in the horizontal DFT androtation units, where the selection of radix-2*k* algorithm makes a difference to the overall consumption

$$\lambda_c^g = \begin{cases} \frac{P}{2} \cdot \log_2\left(\frac{N}{P}\right) - 1, & \text{for radix-2} \\ P \cdot \left\lceil \log_{2^k}\left(\frac{N}{P}\right) \right\rceil - 1, & \text{for radix-}2^k (k > 1). \end{cases} \tag{11}$$

It can be found from (11) that high radices are beneficialto reduce the general complex multipliers at the expenseof increasing the number of constant complex multipliers.Apart from the $P\_\log 2k\ (N/P)$ constant multipliers used to implement the horizontal DFT unit, additional ones will be

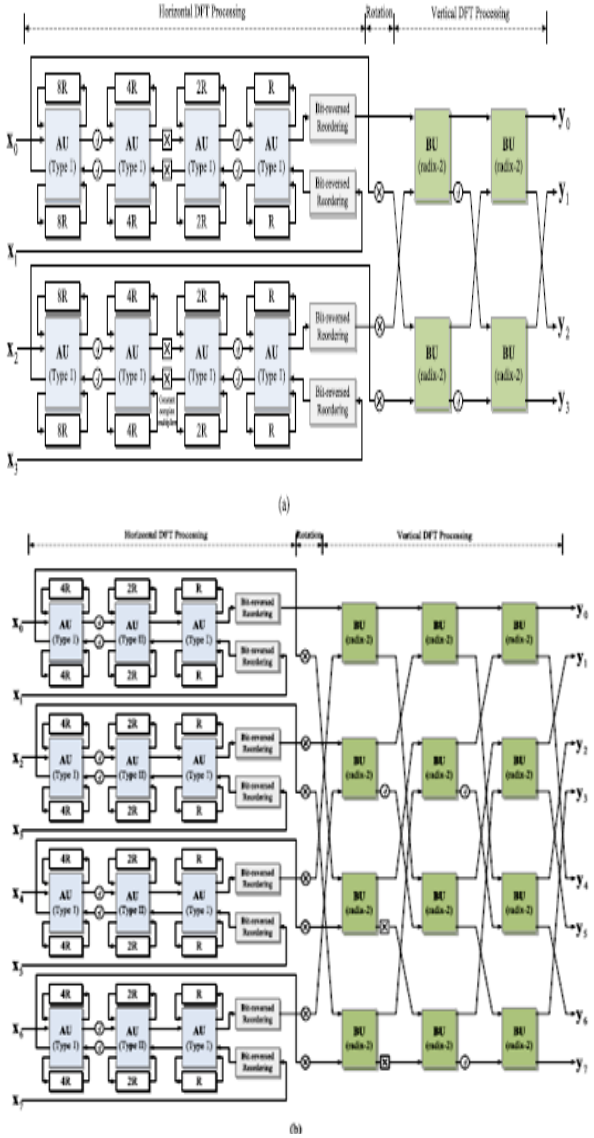

Fig.6. Proposed M2DF architectures for the computation of 64-point DFT. (a) Four-parallel computation, use radix-24 algorithm in the horizontal DFT processing phase. (b) Eight-parallel computation, use radix-23 algorithm in the horizontal DFT processing phase.

Consumed by vertical DFT unit when $P > 8$. In terms of thememory consumption, $N - P$ registers applied to implementcontrol circuit, together with the additional $N + P$ onesto perform the reordering operation are considered in thefollowing comparison.Table I compares the proposed

structures with other efficientapproaches in the two-parallel, four-parallel, and eight-parallelscenario, respectively. In spite of the differences among underlyingdesign, we suppose the input and output parallel streamsof different kinds of DFT processors follow the definitionsin (7a) and (7b). As shown in the table, one of the remarkablefeatures of the M2DF structures is embodied in the effectiveuse of complex adders. For conventional MDF schemes, theutilization ratio of adders is ∼50%, and the parallelizationcannot improve this ratio. Nevertheless, since the improvement is premised on the preservation of low memory requirement and shortcomputing delay, the MDC approach will surpass the proposedmethod at this metric in certain applications. In this respect,the M^2DF design and the MDC scheme have their own merits.

B. Experimental Results

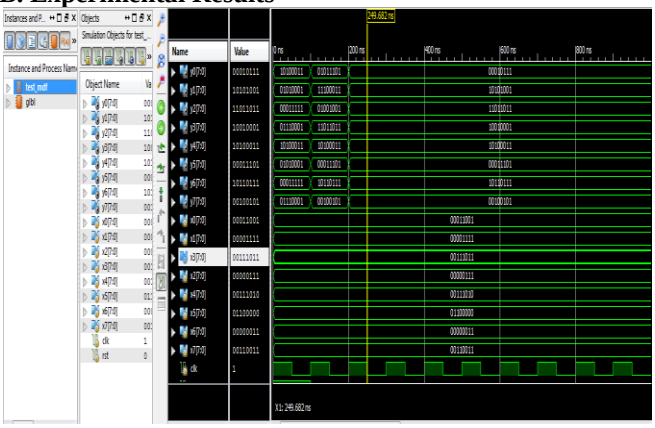

Fig7. Simulation result for the proposed system.

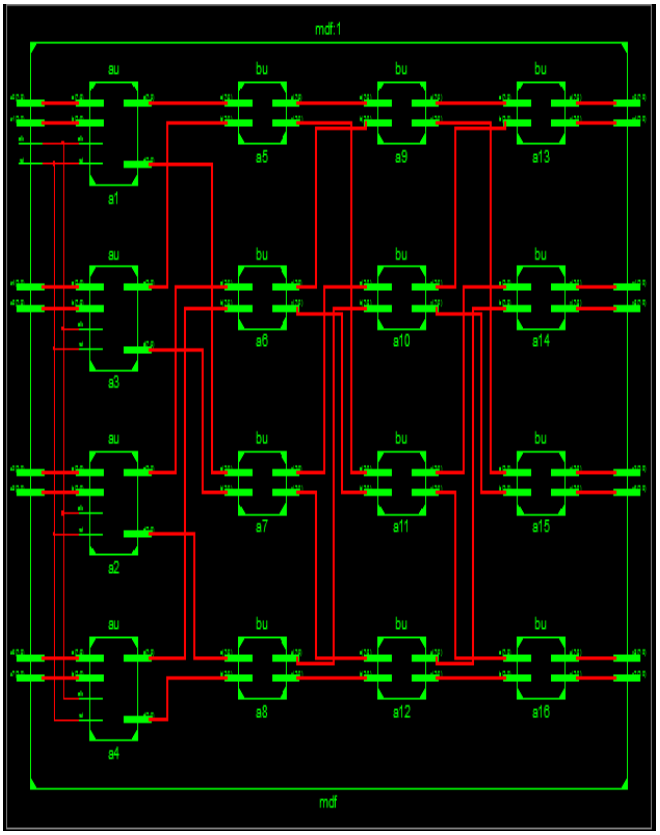

Fig8. RTL schematic view for the proposed system.

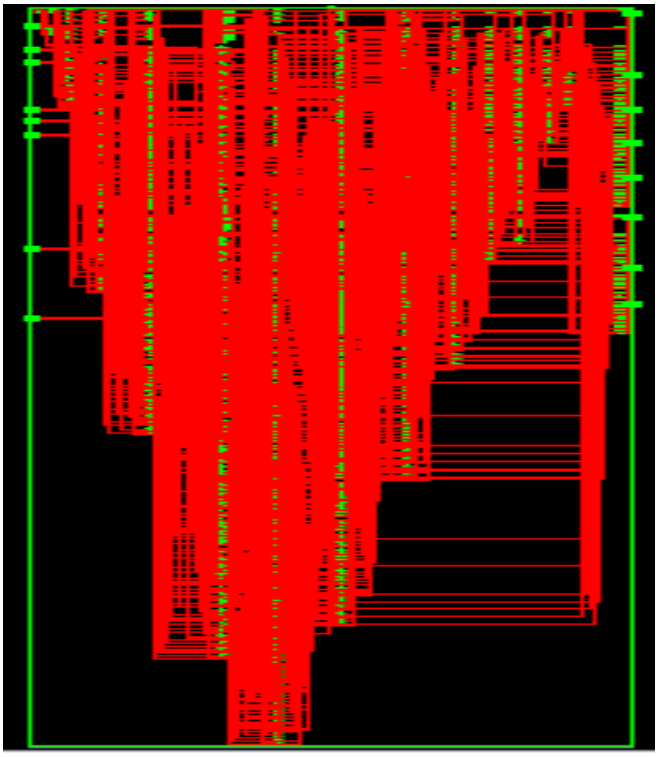

Fig9. Technology schematic view for the proposed system

IV. CONCLUSION

In spite of the low memory requirement and short computing delay, conventional MDF schemes suffer from the inefficient use of adders and multipliers in practical applications. This paper recommends an M2DF structure to grapple with this obstacle, which eliminates the standby time of arithmetic modules in feedback architectures by integrating DIT operations into the DIF-operated computing units. According to the theoretical analysis and experimental results, the M2DF design inherits the strengths of feedback structures while significantly curbing the overexploitation of arithmetic resources. This outstanding feature enables the M^2DF structure to be an efficient alternative to the MDF scheme. On the other hand, the M2DF and the MDC scheme consume the same amount of adders, while they have their own merits in multiplier overhead. However, when the computing concerns, the M2DF approach will be more hardware friendly than the MDC scheme.

V. REFERENCES

[1] G. Bi and E. V. Jones, "A pipelined FFT processor for word-sequentialdata," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12,pp. 1982–1985, Dec. 1989.

[2] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipelineFFT processors for VLSI implementations," *IEEE Trans. Comput.*,vol. C-33, no. 5, pp. 414–426, May 1984.

[3] S. He and M. Torkelson, "Design and implementation of a 1024-pointpipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.*,May 1998, pp. 131–134.

[4] C.-C. Wang, J.-M.Huang, and H.-C. Cheng, "A 2K/8K mode small-areaFFT processor for OFDM demodulation of

DVB-T receivers," *IEEETrans. Consum.Electron.*, vol. 51, no. 1, pp. 28–32, Feb. 2005.
[5] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM(de)modulation," in *Proc. URSI Int. Symp. Signals, Syst., Electron.*,Sep./Oct. 1998, pp. 257–262.
[6] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A new VLSI-oriented FFTalgorithm and implementation," in *Proc. 11th Annu. IEEE Int. Adv. Syst.Integr. Circuits Conf.*, Rochester, NY, USA, Sep. 1998, pp. 337–341.
[7] T. Lenart and V. Öwall, "A 2048 complex point FFT processor usinga novel data scaling approach," in *Proc. IEEE Int. Symp. CircuitsSyst. (ISCAS)*, Bangkok, Thailand, May 2003, pp. IV-45–IV-48.
[8] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT,"*IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874, Mar. 2003.
[9] C. Cheng and K. K. Parhi, "High-throughput VLSI architecture for FFTcomputation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 10,pp. 863–867, Oct. 2007.
[10] Y.-N. Chang, "An efficient VLSI architecture for normal I/O orderpipeline FFT design," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55,no. 12, pp. 1234–1238, Dec. 2008.
[11] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFTarchitectures via folding transformation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.
[12] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson,"Pipelinedradix-2*k* feedforwardFFT architectures, " *IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.
[13] K.-J. Yang, S.-H. Tsai, and G. C. H. Huang, "MDC FFT/IFFT processorwith variable length for MIMO-OFDM systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 4, pp. 720–731, Apr. 2013.
[14] J. Lee, H. Lee, S.-I. Cho, and S.-S. Choi, "A high-speed, low-complexityradix-24 FFT processor for MB-OFDM UWB systems," in *Proc. IEEEInt. Symp. Circuits Syst.*, May 2006, pp. 210–213.
[15] S.-I. Cho, K.-M.Kang, and S.-S. Choi, "Implemention of 128-point fastFourier transform processor for UWB systems," in *Proc. Int. WirelessCommun. Mobile Comput. Conf.*, Aug. 2008, pp. 210–213.
[16] H. Liu and H. Lee, "A high performance four-parallel 128/64-pointradix-24 FFT/IFFT processor for MIMO-OFDM systems," in *Proc.IEEE Asia Pacific Conf. Circuits Syst.*, Nov./Dec. 2008, pp. 834–837.
[17] N. Li and N. P. Van Der Meijs, "A radix-22 based parallel pipeline FFTprocessor for MB-OFDM UWB system," in *Proc. IEEE Int. SOC Conf.*,Sep. 2009, pp. 383–386.
[18] S.-N. Tang, J.-W.Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT processor forOFDM-based WPAN applications," *IEEE Trans. Circuits Syst. I, Reg.Papers*, vol. 57, no. 6, pp. 451–455, Jun. 2010.
[19] C.-H. Yang, T.-H. Yu, and D. Markovic, "Power and area minimizationof reconfigurable FFT processors: A 3GPP-LTE example," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 757–768, Mar. 2012.
[20] S.-N. Tang, C.-H.Liao, and T.-Y. Chang, "An area- and energy-efficientmultimode FFT processor for WPAN/ WLAN/ WMAN systems," *IEEEJ. Solid-State Circuits*, vol. 47, no. 6, pp. 1419–1435, Jun. 2012.
[21] T. Cho and H. Lee, "A high-speed low-complexity radix-25 modifiedFFT processor for high rate WPAN applications," *IEEE Trans. VeryLarge Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 187–191, Jan. 2013.
[22] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design andImplementation*. Hoboken, NJ, USA: Wiley, 1999.
[23] A. Cortes, I. Velez, and J. F. Sevillano, "Radix *rk*FFTs: MatricalrepresentationandSDC/SDFpipelineimplementation, " *IEEE Trans.Signal Process.*, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

Author's Profile:

Sk.Karishma received her B.tech degree in Electronics & Communication Engineering from Audisankara College of Engineering for Women, Gudur, SPSR Nellore District, affiliated to JNTU Anantapuram. She is currently pursuing M.Tech VLSI in Audisankara College of Engineering and Technology Gudur (Autonomous), SPSR Nellore(Dist), affiliated to JNTU Anantapuram.

Prof. K.Dhanunjaya received his B.Tech Degree in Electronics & Communication Engineering from G.Pulla Reddy Engineering College, Kurnool, AP in 1998, and M.Tech in ECE from Jawaharlal Nehru Technological University Kakinada in 2001. He is currently perusing PhD in Low power VLSI design from Jawaharlal Nehru Technological University, Anantapuram. He has 17 years teaching experience, presently working as professor & Head of the department of ECE, Audisankara Engineering and Technology (Autonomous), Affiliated to College of JNTUA, Gudur. He is a life time member of IETE & ISTE and member of IEEE.