



## **MDF ARCHITECTURE DESIGN FOR RADIX 2K FFT**

**D. Sathyakala\* & S. Nithya\*\***

\* Assistant Professor, Department of ECE, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

\*\* Assistant Professor, Department of ECE, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

### **Abstract:**

*This paper presents a mixed-decimation multipath delay feedback ( $M^2DF$ ) approach for the radix-2k fast Fourier transform. Here, principle of folding transformation to derive is the proposed architecture, which initiates the idle period of arithmetic modules in multipath delay feedback (MDF) architectures via integrating the decimation-in-time operations into the decimation-in-frequency-operated computing units. Furthermore, we relate the proposed design with other efficient schemes, namely, the MDF and the Multipath Delay Commutator (MDC) scheme theoretically and experimentally. It can be decided that the  $M^2DF$  design assists as an efficient alternative to the MDF scheme, since it reaches improved efficiency in the utilization of arithmetic resources without deteriorating the superiorities of feedback structures. In addition, the suggested design performs better in memory requirement and computing delay.*

**Key Words:** Decimation-In-Frequency (DIF), Decimation-In Time (DIT), Fast Fourier Transform (FFT), Multipath Delay Feedback (MDF) & Pipelined-Parallel Architecture

### **1. 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-time business, plenty of published works have focused on designing and 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 most classical approaches to perform the pipelined FFT computation. To reduce the memory banks in SDC pipelines, Single-path Delay Feedback (SDF) architecture is proposed in [2], which is characterized by the feedback connections in the circuits. These hardware schemes can be combined with the radix-2 [2], [3], radix-4 [4], and especially radix-2k algorithm [5]–[8] to execute the DFT operation. Compared with the radix-4 approach, the radix-2k pipeline is equipped with simpler butterfly units while making better utilization of complex multipliers than the typical radix-2 scheme.

Thus, radix-2k algorithm acts as an effective alternative to the conventional computation methods from the perspective of hardware design. Admittedly, the extension of communication service has stimulated dramatic rise of throughput requirements. This advance in real-time business has additionally rendered the hardware schemes designed for SISO applications obsolete to a certain extent. On this occasion, Multipath Delay Commutator (MDC) [9]–[13] and Multipath Delay Feedback (MDF) [14]–[21], which serves as the upgrade of SDC and SDF, respectively, are

proposed to calculate the FFT when several samples of the same sequence are received in parallel. In general, the MDF structure is composed of multiple interconnected SDF paths, and each path is responsible for managing one of the parallel input streams. This design contributes to the inheritance of utilizing registers efficiently at the expense of squandering the arithmetic components, particularly the butterfly units. By contrast, the MDC approach paves the way for boosting the hardware efficiency of arithmetic units (AUs), while additional memories have to be consumed or either reordering the samples or folding the streams, which additionally leads to an increase of computing delay. Whether feed forward structures represented by the MDC approach or feedback structures, such as the SDF and the MDF design, they afford feasible solutions to strike a balance between the consumption of hardware resources and the reachable performance.

As we move forward the discussion, the hardware resources can be further divided into two categories:

- Arithmetic resources associated with logical or arithmetic operations and
- Memory resources referred to as memory units, which are responsible for caching samples.

Due to the outstanding performance in the efficient use of memory resources, the MDF scheme has been a tremendous success in a variety of applications [17]–[21]. Underneath the triumph, the underutilization of arithmetic resources is still a stubborn problem for feedback design and has not been resolved satisfactorily. The objective of this paper is to make a breakthrough in this respect while maintaining the advantages of feedback structures. To this end, the theory of folding transformation described in [22] and first adopted to derive the pipelined FFT architectures in [11] is employed to derive the proposed scheme, namely, the mixed-decimation MDF(M<sup>2</sup>DF) architecture. The kernel of M<sup>2</sup>DF design lies in scheduling the decimation-in-time (DIT) operations onto the decimation-in-frequency (DIF)-operated basic blocks. By this means, we mobilize the idle period of arithmetic modules in MDF architectures and thereby gain a considerable reduction of arithmetic resources to make up for the deficiency of the feedback FFT modules. The rest of this paper is organized as follows. In Section II, we introduce the top-level architecture of the pipelined-parallel FFT processor based on the matrix factorization of radix-2 $k$  algorithm. Section 3 expands on the M<sup>2</sup>DF architectures using folding transformation. Theoretical and experimental comparisons of the M<sup>2</sup>DF scheme and existing approaches are related in Section 4. Finally, the conclusion is drawn in Section 5.

## **2. Design of Mixed-Decimation MDF Architecture:**

In this section, we first apply the theory of folding transformation to derive the folding matrices associated with DIF- and DIT-operated SDF processor. With the assist of this theoretical basis, the operations in SDF pipeline are rescheduled to reverse the underutilization of arithmetic modules, which is accomplished by integrating DIT operations into DIF-operated basic blocks. Finally, this optimization strategy is used to implement the top-level design in Section II, generating the proposed M<sup>2</sup>DF structure with improved hardware efficiency.

### **A. Analyzing the SDF Scheme using Folding:**

**Transformation:** The folding transformation provides a systematic technique for designing control circuits for hardware where several algorithm operations are time multiplexed on a single computing device [22]. Consider an eight-point radix-2 $k$ DIF computation; the algorithm can be presented using a data flow graph shown in Fig. 2(a), where the nodes represent computations and the directed edges represent data paths.

As shown, the flow graph consists of three stages and four operations are executed within each stage. When  $x_i, i = 0, \dots, 7$  arrives in serial, multiple operations in each stage can be time multiplexed to a single computing unit without any collisions, and the control circuits are determined systematically by folding transformation. In this way, the FFT flow graph in Fig. 2(a) can be transformed into a pipelined form shown in Fig. 2(b), where operations  $A_0, \dots, A_3, B_0, \dots, B_3, C_0, \dots, C_3$  are performed in the computing unit  $U_A, U_B,$  and  $U_C,$  respectively. The multiple operations included in a computing module are arranged by folding sets. A folding set is an ordered set of operations executed by the same computing unit. Apart from the operations associated with the nodes in the dataflow graph; generally, there are also null operations in the folding set. The number of operations folded into a folding set is called the folding factor (denoted by  $R$  in subsequent discussion). Correspondingly, the computing unit works in cycles with a period taking up  $R$  time partitions, and the operation in the  $r$ th ( $r = 0, \dots, R - 1$ ) position is executed.

During  $r^{\text{th}}$  time partition. For example, consider a folding set  $F = \{\varphi A_0 \varphi A_1 \varphi A_2 \varphi A_3\}$  with  $R = 8$ , the computing unit operates  $A_0, A_1, A_2,$  and  $A_3$  in the first, third, fifth, and seventh time partitions, respectively, while keeping idle in remaining intervals. When folding transformation is utilized to derive the circuit from a data flow graph, the accurate determination of folding sets serves as the cornerstone. In particular, the folding factor depends on the periodicity of operations executed by a computing unit, while the arrangement of elements within a folding set is linked closely to specific hardware structures. In terms of an  $N$ -point SDF processor using a radix- $2k$  DIFFFT algorithm, it constitutes of  $\log_2 N$  computing modules. The  $l$ th ( $l = 0, 1, \dots, \log_2 N - 1$ ) module alternates idle condition and operating status with an interval, including  $N/2^{l+1}$  time instances.

Taking this prior knowledge into account, the folding sets corresponding to  $U_A, U_B,$  and  $U_C$  in Fig. 2(b) can be written preliminarily as

$$\tilde{F}_A = \{\varphi \varphi \varphi \varphi A_0 A_1 A_2 A_3\} \dots \dots \dots (1a)$$

$$\tilde{F}_B = \{\varphi \varphi B_0 B_1 \varphi \varphi B_2 B_3\} \dots \dots \dots (1b)$$

$$\tilde{F}_C = \{\varphi C_0 \varphi C_1 \varphi C_2 \varphi C_3\} \dots \dots \dots (1c) \text{ with the folding factor } R=8.$$

Equations (1a)-(1c) incorporate the features of SDF computing units into the derivation of folding sets. Nevertheless, they fail to consider the precedence of operations associated with different modules. As can be found in Fig. 2(b), when the data stream flows serially to  $U_A$ , the newly generated intermediate results would arrive at  $U_B$  and  $U_C$  with a delay of four and six time partitions, respectively, after which the modules are able to perform the operations formulated by (1b) and (1c). In view of this causality constraint, the  $m$ th ( $m = 0, \dots, R-1$ ) element in  $\tilde{F}_B$  ought to be aligned with the  $[m + 4]$   $R$ th entry of  $\tilde{F}_A$  ( $[\cdot]_x$  returns the modulus of  $x$ ).

Likewise, the  $n$ th ( $n = 0, \dots, R - 1$ ) element in  $\tilde{F}_C$  requires to match the  $[n + 6]$   $R$ th position of  $\tilde{F}_A$ . This yields modified folding sets

$$F_A = \{\varphi \varphi \varphi \varphi A_0 A_1 A_2 A_3\}$$

$$F_B = \{\varphi \varphi B_0 B_1 \varphi \varphi B_2 B_3\}$$

$$F_C = \{\varphi C_0 \varphi C_1 \varphi C_2 \varphi C_3\}$$

Where  $F_A$  inherits  $\tilde{F}_A$  directly. By contrast,  $F_B, F_C$  can be obtained by cyclic right shifting the elements in  $\tilde{F}_B, \tilde{F}_C$  for four and six positions, respectively. As folding sets reveal limitations when analyzing the entire hardware structure, we commend an alternative analytical tool named folding matrix to cover this shortage. Denote an  $n_f \times R$  matrix

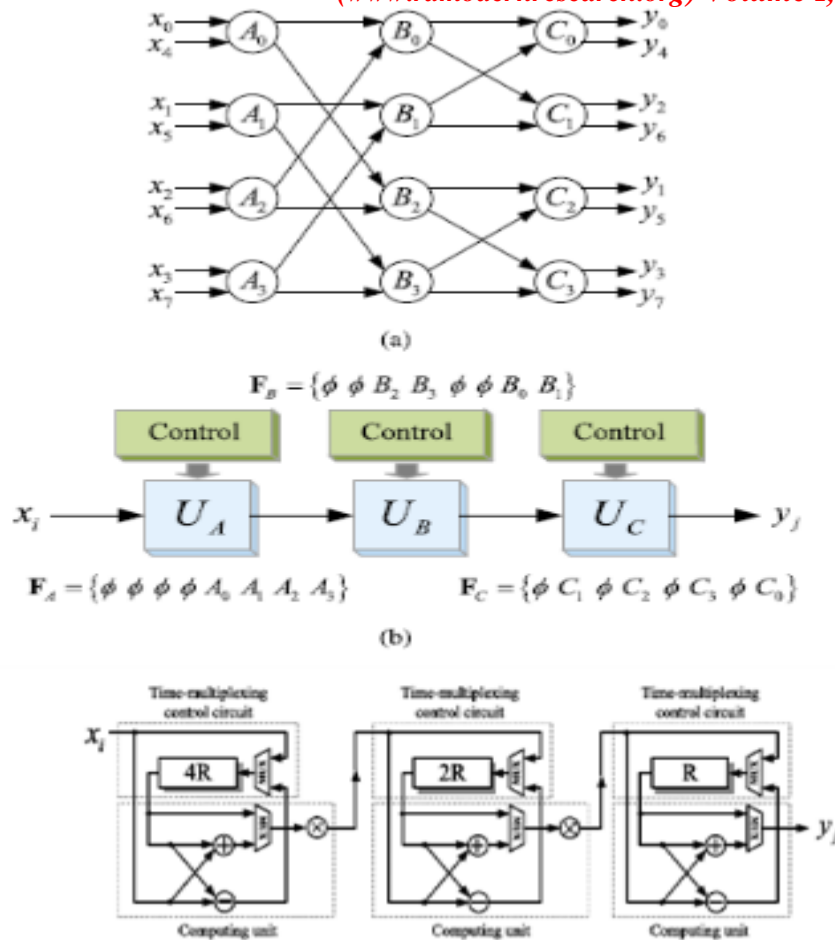


Figure 1: (a) Data flow graph of an eight-point DIF DFT using a radix-2k algorithm. (b) Data flow graph can be converted into a pipelined version through folding transformation. (c) Optimized hardware scheme to implement the pipelined flow graph. Moreover, the shift register unit marked n R indicates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

$$F_{DIF} = \begin{bmatrix} \phi & \phi & \phi & \phi & A_0 & A_1 & A_2 & A_3 \\ \phi & \phi & B_2 & B_3 & \phi & \phi & B_0 & B_1 \\ \phi & C_1 & \phi & C_2 & \phi & C_3 & \phi & C_0 \end{bmatrix} \dots \dots \dots (3)$$

as the folding matrix corresponding to (1a)-(1c), where  $n f$  equals the amount of computing stages in the folded flow graph, while the rows of  $F_{DIF}$  corresponds to different folding sets. Therefore, the entry  $(F_{DIF})m, n(m = 0, \dots, n f - 1, n = 0, \dots, R - 1)$  represents the operation executed by the  $m$ th computing unit at the  $l R + n$ th ( $l = 1, 2, \dots$ ) time instance. When obtaining the folding sets associated with  $U_A$ ,  $U_B$ , and  $U_C$ , the supporting circuits used to fulfill the time multiplexing of computing units can be obtained through folding transformation. To boost the hardware efficiency, the implementation scheme can be further optimized by introducing the register minimization technique, which eventually generates the SDF hardware structure presented in Fig. 2(c). The shift register unit marked  $n R$  in the graph indicates that  $n$  positions are available in the component 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 transposition of its DIF counterpart. Assuming that the input samples arrive in bit-reversed order, the folding sets considering the property of SDF computing are

$$\bar{G}_A = \{\phi \bar{A}_0 \phi \bar{A}_0 \phi \bar{A}_1 \phi \bar{A}_3\}$$

$$\begin{aligned} \bar{G}_B &= \{\varphi \varphi \bar{B}_0 \bar{B}_1 \varphi \varphi \bar{B}_2 \bar{B}_3\} \\ \bar{G}_C &= \{\varphi \varphi \varphi \varphi \bar{C}_0 \bar{C}_1 \bar{C}_2 \bar{C}_3\} \dots\dots\dots(4) \end{aligned}$$

With the folding factor  $R = 8$ . Note that the arrival of freshly generated results at  $U \sim B$  and  $U \sim C$  involve additional delay of one and three time partitions, thus by cyclic right shifting the

$$\begin{aligned} G_{\bar{A}} &= \{\varphi \bar{A}_0 \varphi \bar{A}_0 \varphi \bar{A}_1 \varphi \bar{A}_3\} \\ G_{\bar{B}} &= \{\bar{B}_3 \varphi \varphi \bar{B}_0 \bar{B}_1 \varphi \varphi \bar{B}_2\} \\ G_{\bar{C}} &= \{\bar{C}_1 \bar{C}_2 \bar{C}_3 \varphi \varphi \varphi \varphi \bar{C}_0\} \dots\dots\dots(5) \end{aligned}$$

Accordingly, the  $nf \times R$  folding matrix corresponding to the DIT data flow graph is expressed as

$$G_{DIT} = \begin{bmatrix} \varphi \bar{A}_0 \varphi \bar{A}_0 \varphi \bar{A}_1 \varphi \bar{A}_3 \\ \bar{B}_3 \varphi \varphi \bar{B}_0 \bar{B}_1 \varphi \varphi \bar{B}_2 \\ \bar{C}_1 \bar{C}_2 \bar{C}_3 \varphi \varphi \varphi \varphi \bar{C}_0 \end{bmatrix} \dots\dots\dots(6)$$

The entry  $(G_{DIT})_m, n(m = 0, \dots, nf - 1, n = 0, \dots, R - 1)$  also represents the operation executed by the  $m$ th computing module at the  $lR + nth$  ( $l = 1, 2, \dots$ ) time instance. With the assist of folding transformation and register minimization techniques, the hardware scheme can be derived, which is shown in Figure 3(c).

**B. Rescheduling the Operations in SDF Pipelines:**

The folding matrix given in (3) and (7) conveys the fact that whether designers adopt the DIF approach or DIT scheme to construct the SDF circuit, the existence of null operations in folding sets will always degrade the efficiency of arithmetic components, leading to an approximate 50% utilization of complex adders merely. To address this issue, we reschedule the operations in SDF pipelines to activate the idle intervals of computing units. As the folding matrix acts as the simplified representation of circuit status, the rearrangement of operations can be achieved equivalently by modifying the folding matrices. To be specific, the modification is accomplished through the following three steps.

Cyclic right shift the columns of  $G_{DIT}$  for  $SR = 1$  space, it generates

$$(G_{DIT})_1 = \begin{bmatrix} \bar{A}_3 \varphi \bar{A}_0 \varphi \bar{A}_0 \varphi \bar{A}_1 \varphi \\ \bar{B}_2 \bar{B}_3 \varphi \varphi \bar{B}_0 \bar{B}_1 \varphi \varphi \\ \bar{C}_0 \bar{C}_1 \bar{C}_2 \bar{C}_3 \varphi \varphi \varphi \varphi \end{bmatrix} \dots\dots\dots(7)$$

where the subscript outside the brackets denotes the cyclic right shift operation. Note the transformation introduces an additional  $SR$ -unit delay to the folding sets  $G \sim A, G \sim B$  and  $G \sim C$  simultaneously, thus the entry  $[(G_{DIT})_1]m, n(m = 0, \dots, nf - 1, n = 0, \dots, R - 1)$  represents the operation executed by the  $m$ th computing module at the  $lR + n - 1$ th ( $l = 1, 2, \dots$ ) time instance, differing from the definitions of  $G_{DIT}$ . To fulfill the operations formulated by (7), a latency unit providing a  $SR$ -clock-cycle delay for input samples should be integrated into the circuit associated with  $G_{DIT}$ . Flip  $(G_{DIT})_1$  vertically to obtain

$$(G_{DIT})_1^F = \begin{bmatrix} \bar{C}_0 \bar{C}_1 \bar{C}_2 \bar{C}_3 \varphi \varphi \varphi \varphi \\ \bar{B}_2 \bar{B}_3 \varphi \varphi \bar{B}_0 \bar{B}_1 \varphi \varphi \\ \bar{A}_3 \varphi \bar{A}_0 \varphi \bar{A}_0 \varphi \bar{A}_1 \varphi \end{bmatrix} \dots\dots\dots(8)$$

Where the added superscript stands for the vertical flip. As the rows of folding matrix are connected with specific computing units, the modification of current step would redefine this mapping relation, i.e., the  $m$ th ( $m = 0, \dots, nf - 2$  row of  $(G_{DIT})_1^F$  describes the time-multiplexing scheme of the  $nf - m - 1$ th computing unit, rather than the  $m$ th module as the original  $G_{DIT}$ .



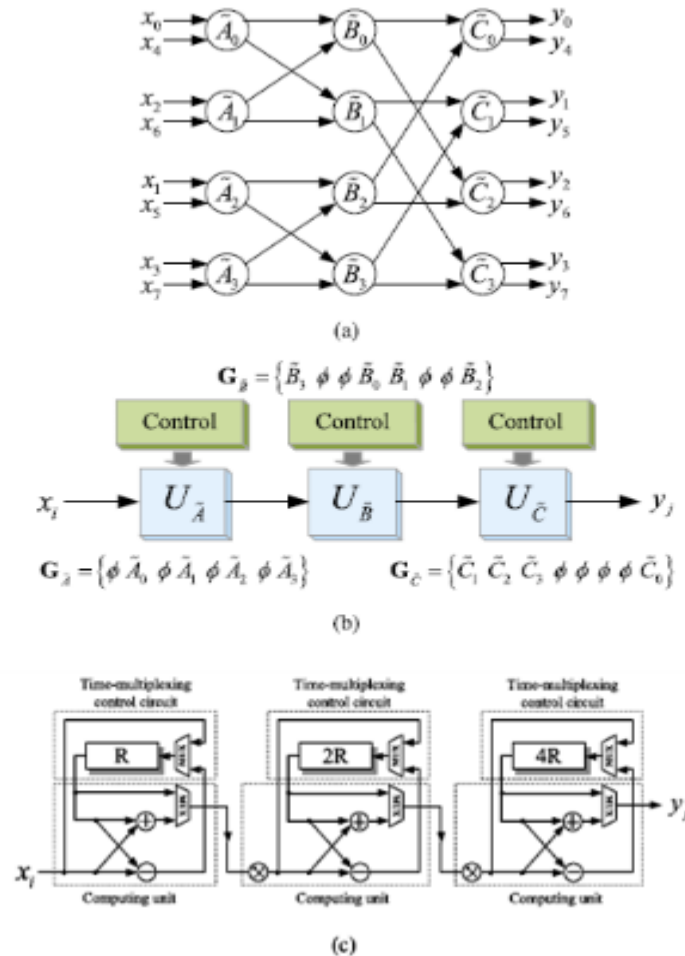


Figure 2: (a) Data flow graph of an eight-point DIT DFT using a radix-2k algorithm. (b)

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  $\tilde{G}^B$  for one space and  $\tilde{G}^C$  for three spaces, we obtain. Overlay  $F_{DIF}$  with  $(G_{DIT})^F$

$$\text{Thus, we can obtain } F = F_{DIF} + (G_{DIF})_1^F = \begin{bmatrix} \tilde{C}_0 \tilde{C}_1 \tilde{C}_2 \tilde{C}_3 A_0 A_1 A_2 A_3 \\ \tilde{B}_2 \tilde{B}_3 B_2 B_3 \tilde{B}_0 \tilde{B}_1 B_0 B_1 \\ \tilde{A}_3 C_1 \tilde{A}_0 C_2 \tilde{A}_0 C_3 \tilde{A}_1 C_0 \end{bmatrix} \dots \dots \dots (9)$$

Clearly, the superposition eliminates the pre existing null operations, which suggests that the synthesis of DITSDF pipeline and DIF SDF pipeline is able to fully utilize the arithmetic modules. With respect to the hardware implementation, as shown in Fig. 4, the  $m$ th ( $m = 0, \dots, n f - 1$ ) computing unit in the DIF SDF pipeline should be integrated with then  $f - m - 1$ th one in the DIT counterpart due to the vertical flip operation in the second step. In addition, the DIT-processing stream ought to experience a  $SR$ -clock cycle delay up front

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 the previous discussion, there are a set of candidates

$$S_R = nR + 1, n \in N \dots \dots \dots (10)$$

With  $R$  represents the folding factor, which is equal to the FFT size in serial computing. In terms of the underlying design of arithmetic modules, the conventional SDF elementary structure should be upgraded to undertake both DIF processing and DIT is processing tasks. According to the utilization efficiency of complex multipliers,

two kinds of hardware structures referred to as Type I and Type II are proposed in Fig. 5. For Type I architecture in Fig. 5(a), adders are shared by two streams, while multipliers (not exhibited) are provided individually.

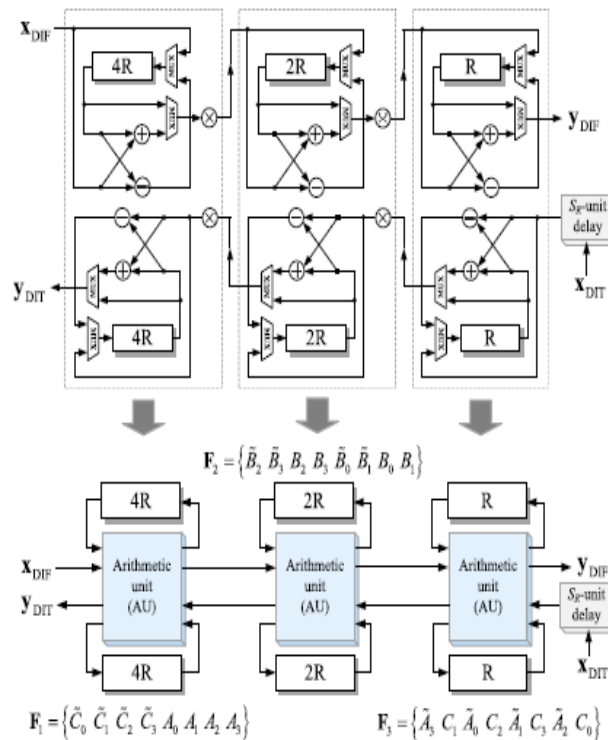


Figure 3: Integration of the DIF and DIT pipelined processor to fully utilize the computing units

Type II architecture is shown in Figure 5(b), where both adders and multipliers are reused to further improve the hardware efficiency. Thus, Type II is suitable for the situation when data streams occupy the complex multipliers with a 50% utilization ratio or less. Moreover, the multiplexors and selectors in the circuit can be divided into two categories (white for Category I and gray for Category II in Figure 5. When arithmetic modules are in service, the components within each category share the identical logic signal. The control scheme, which is synchronized with the DIF-computing stream, is summarized as follows.

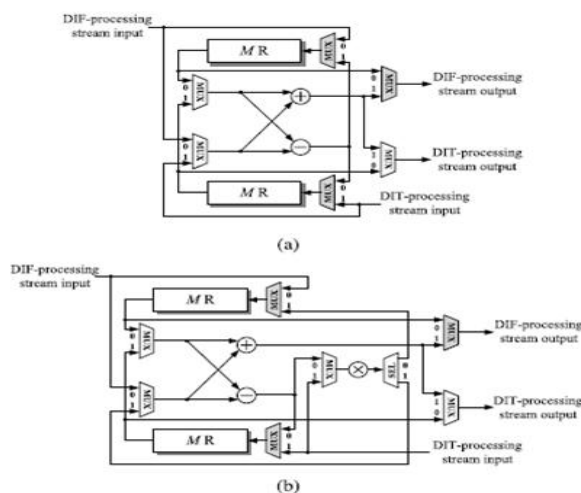


Figure 4: Underlying structures of AUs. (a) Type I structure, complex adders are shared only. (b) Type II structure, both complex adders and complex multipliers are shared

- For the first  $M$  samples of the stream, the components belonging to Category I are controlled by logic 1, while others in Category II are controlled by logic 0.
- During the next  $M$  samples, the control signals should be inverted.

The scheme above is suitable for both Type I and Type II design and is  $2M$ -clock-cycle periodic, where  $M$  is the length of shift register sets in the arithmetic module.

### C. Design of the M<sup>2</sup>DF Hardware Structure:

For the top-level design shown in Fig1, the rescheduling strategy proposed in Section III-B is beneficial to optimize the underlying scheme, which results in the proposed M<sup>2</sup>DF structure. We begin with two-parallel architecture to unfold the kernel. Taking a 32-point DFT computation, for example, the corresponding M<sup>2</sup>DF 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  $x_0TS$  and  $x_1TS$ . By drawing on the discussion in Section III-B, the pipelined circuit constructed by the modified AUs in Fig. 5 is able to fulfill this task, which promotes the arithmetic resources to full utilization. In addition, as input samples require to be rearranged as bit reversal before participating in DIT computation, the latency unit in Fig. 4 is replaced with an ordering module, which permutes the samples with a delay of  $SR = S + 1$  clock cycles. After the rotation, the parallel streams are connected to a simple radix-2 butterfly unit to execute the vertical DFT processing. As for the four-parallel and eight-parallel applications, Fig. 7 shows the hardware structures using a 64-point DFT. As shown, the circuit used to perform the horizontal DFT is essentially a parallel extension of the counterpart in the two-parallel structure. By contrast, the hardware scheme for vertical DFT depends closely on the parallelism. For four-parallel computing in Fig. 7(a), four radix-2 butterfly units along with requisite rotation devices operate coordinately to provide a throughput of four samples per clock cycle.

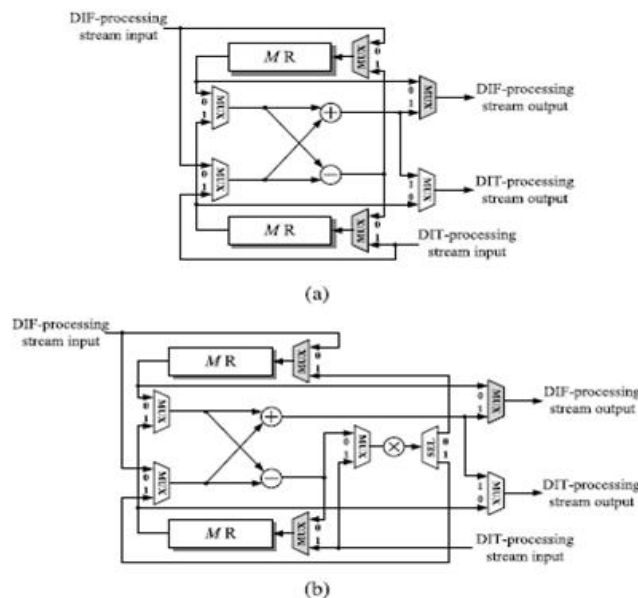


Figure 5: Proposed two-parallel M<sup>2</sup>DF architectures for the computation of 32-point DFT. (a) Use radix-2 algorithm in the horizontal DFT processing phase. (b) Use radix-22 algorithm in the horizontal DFT processing phase

In the eight-parallel scenario, by means of rearranging the interconnections of the eight-point radix-2 data flow graph, the multiple stages included in the vertical DFT circuit can be implemented using a fixed structure, which contributes to reduce the hardware complexity. Furthermore, note each multiplier in the vertical DFT unit is



related to a single known coefficient, they can be mapped to the low-complexity constant multipliers. The M<sup>2</sup>DF structure can be readily applied to the  $P$ -parallel radix- $2k$  computation, where even  $P$  is emphasized here. The exponent  $k$  will only affect the arrangement of complex multipliers in the horizontal DFT circuit. On the other hand, the parallelism  $P$  plays an important role in the realization of vertical DFT unit, as this module is responsible for a  $P$ -parallel  $P$ -point DFT. In terms of the rotation,  $P - 1$  complex multipliers are sufficient to complete the relevant task, which are irrelevant to the selection of radix- $2k$  Algorithm.

### **3. Comparison and Evaluation:**

In this section, we first evaluate the hardware expense of M<sup>2</sup>DF structures. Afterward, we compare the M<sup>2</sup>DF scheme with existing approaches theoretically and experimentally.

#### **A. Evaluation of the Proposed M<sup>2</sup>DF 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 component is primarily composed of complex adders and multipliers to execute the butterfly operations and twiddle factor multiplications. Moreover, the multipliers can be further categorized into the general ones with varied twiddle factors and constant ones using fixed coefficients. This fine-grained partition takes account of the diverse hardware complexity between general complex multipliers and constant ones. In this way, the numbers of complex adders  $\lambda^a$ , general complex multipliers  $\lambda^g$  and constant complex multipliers account for the primary consumption of arithmetic resources in an FFT module. Memory expenditure comes from the realization of control circuit and reordering module. The control circuit is responsible for the rearrangement of data streams to coordinate with the arithmetic computations. In pipelined structures, shift register sets are widely used to implement the control circuit. On the other hand, the reordering module permutes the samples to obtain the desired parallel output  $\text{unvec}_{P,S}(y)$ . To this end,  $Y$  in (7b) should be gained firstly, after which the permutation network corresponding to the matrix  $P(P)N$  is applied to bridge the gap between  $Y$  and  $\text{unvec}_{P,S}(y)$ . It is worth noting that the hardware design of  $P(P)N$  could be identical for the FFT processors with the same parallelism, while the circuit to obtain  $Y$  is linked closely to concrete FFT schemes.

Thus, in the following analysis, the memory cost to complete the first-phase reordering is considered as the metric to evaluate the hardware efficiency. Furthermore, some FFT processors do not utilize  $\text{unvec}_{P,S}(x)$  directly, e.g., Garrido *et al.* [12] Convert  $\text{unvec}_{P,S}(x)$  into  $[\text{unvec}_{S,P}(x)]^T$  to launch the FFT. This additional memory requirement is also considered here. For the M<sup>2</sup>DF design,  $P/2 \cdot \log_2 N$  radix-2 butterfly units, which consist of  $\lambda^a = P \cdot \log_2 N$  complex adders are an integral part of the  $P$ -parallel,  $N$ -point DFT computation. The general Complex multipliers are existed in the horizontal DFT and rotation units, where the selection of radix- $2k$  algorithm makes a difference to the overall consumption

$$\lambda_c^g = \left\{ \begin{array}{l} \frac{P}{2} \log_2 \left( \frac{N}{P} \right) - 1, \text{ for radix} - 2 \\ P \left[ \log_{2^k} \left( \frac{N}{P} \right) \right] - 1, \text{ for radix} - 2^k (k > 1) \end{array} \right\} \dots (11)$$

It can be found from (11) that high radices are beneficial to reduce the general complex multipliers at the expense of increasing the number of constant complex multipliers. Apart from the  $P \cdot \log_2 k (N/P)$  constant multipliers used to implement the horizontal DFT unit, additional ones will be Consumed by vertical DFT unit when  $P > 8$ .

In terms of the memory consumption,  $N - P$  registers applied to implement control circuit, together with the additional  $N + P$  ones to perform the reordering operation are considered in the following comparison. Table I compares the proposed structures with other efficient approaches in the two-parallel, four-parallel, and eight-parallel scenario, respectively. In spite of the differences among underlying design, we suppose the input and output parallel streams of different kinds of DFT processors follow the definitions in (7a) and (7b). As shown in the table, one of the remarkable features of the M<sup>2</sup>DF structures is embodied in the effective use of complex adders. For conventional MDF schemes, the utilization ratio of adders is ~50%, and the parallelization cannot improve this ratio. Nevertheless, since the improvement is premised on the preservation of low memory requirement and short computing delay, the M<sup>2</sup>DC approach will surpass the proposed method at this metric in certain applications. In this respect, the MDF design and the MDC scheme have their own merits.

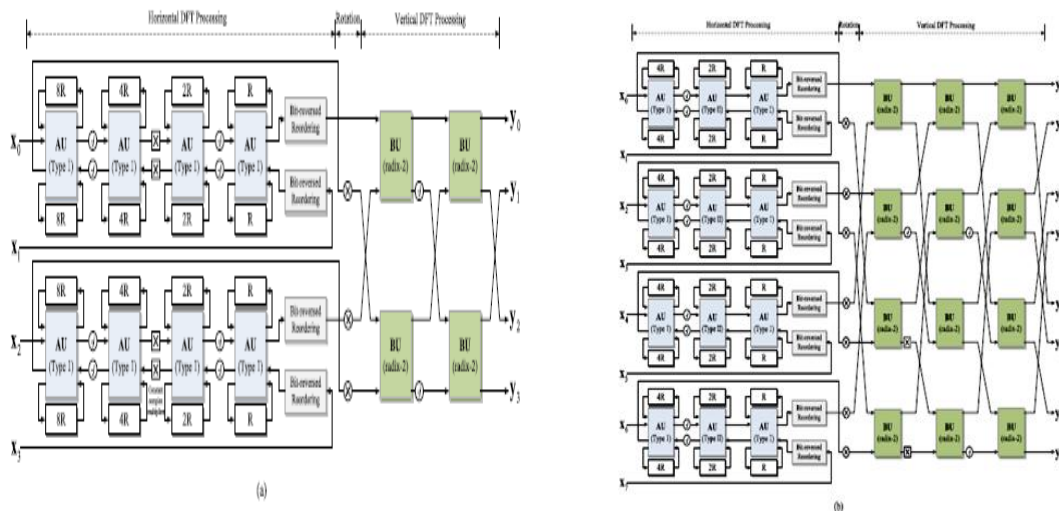


Figure 6: Proposed M2DF architectures for the computation of 64-point DFT. (a) Four-parallel computation, useradix-24 algorithm in the horizontal DFT processing phase. (b) Eight-parallel computation, use radix-23algorithm in the horizontal DFT processing phase

**B. Experimental Results:**



Figure 7: Simulation result for the proposed system

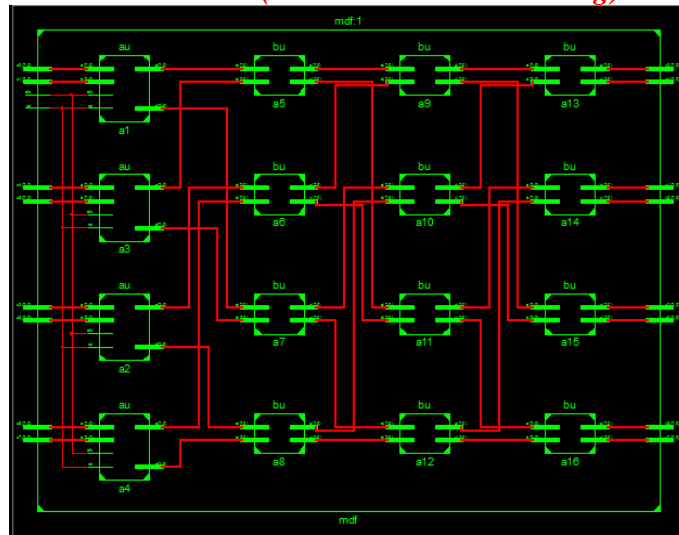


Figure 8: RTL schematic view for the proposed system

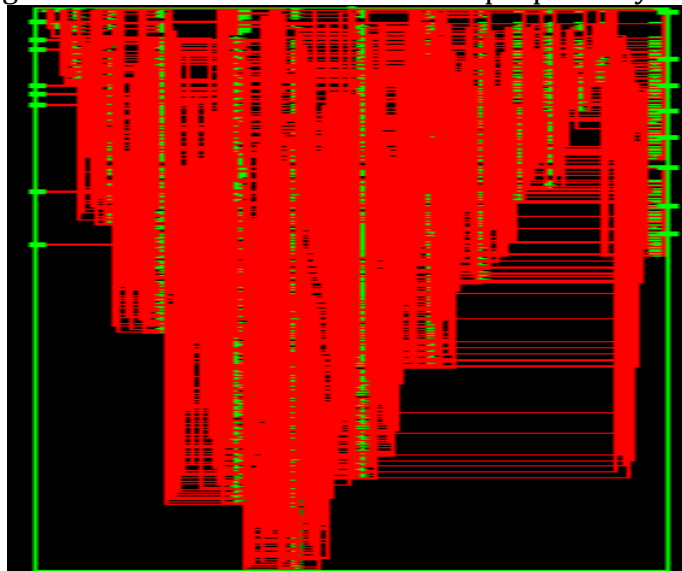


Figure 9: Technology schematic view for the proposed system

#### 4. Conclusion:

The low memory requisite and short computing delay, predictable MDF schemes suffer from the inefficient use of adders and multipliers in practical applications. This paper recommends an M<sup>2</sup>DF structure to struggle with this problem, which rejects the standby time of arithmetic modules in feedback architectures through integrating DIT operations into the DIF-operated computing units. According to the results, the M<sup>2</sup>DF design receives the strengths of feedback structures while significantly curbing the over exploitation of arithmetic resources. This outstanding feature allows the M<sup>2</sup>DF structure to be an efficient alternate to the MDF scheme. On the other hand, the M<sup>2</sup>DF and the MDC scheme consume the same amount of adders, while they have their own merits in multiplier overhead.

#### 5. References:

1. G. Bi and E. V. Jones, "A pipelined FFT processor forward-sequential data," 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 pipeline FFT 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-point pipeline 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-area FFT processor for OFDM demodulation of DVB-T receivers," IEEE Trans. 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 FFT algorithm 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 using a novel data scaling approach," in Proc. IEEE Int. Symp. Circuits Syst. (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 FFT computation," 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 order pipeline 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 FFT architectures 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, "Pipelined radix-2k feed forward FFT architectures," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 1, pp. 23–32, Jan. 2013.
13. K.-J. Yang, S.-H. Tsai, and G. C. H. Huang, "MDCFFT/IFFT processor with 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-complexity radix-24 FFT processor for MB-OFDM UWB systems," in Proc. IEEE Int. Symp. Circuits Syst., May 2006, pp. 210–213.
15. S. I. Cho, K. M. Kang, and S. S. Choi, "Implementation of 128-point fast Fourier transform processor for UWB systems," in Proc. Int. Wireless Commun. Mobile Comput. Conf., Aug. 2008, pp. 210–213.