Low-Complexity Continuous-Flow Memory-Based FFT Architectures for Real-Valued Signals

Jinti Hazarika[1], Mohd Tasleem Khan[2], and Shaik Rafi Ahamed[3]
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
jinti@iitg.ac.in[1], tasleem@iitg.ac.in[2], rafiahamed@iitg.ac.in[3]

***Abstract*—This paper presents two low-complexity continuous-flow memory-based fast Fourier transform (FFT) architectures (Type-I, II) for real-valued signals. Both the proposed designs employ split processing-elements (SPEs), however Type-I SPE processes four inputs while Type-II SPE processes two inputs in parallel. The SPE in Type-I design contains a half-complex multiplier and that of Type-II design contains a quarter-complex multiplier. Two new memory accessing schemes corresponding to each design is proposed. Analysis of computational complexities for both the architectures are carried out and compared with existing designs. It is found that Type-I architecture provides low-complexity in terms of registers and multiplexers while Type-II architecture provides low-complexity in terms of the multiplier. Application specific integrated circuit (ASIC) synthesis and field programmable gate array (FPGA) implementation results show that the proposed designs offer low-area, low-power and utilize less logic elements. For instance, 32-point Type-I real FFT offer requires 5.06% less area, 15.1% less power, 6.58% less sliced look-up table (SLUT) and 5.25% less FF while Type-II 47.76% less area, 43.64% power, 48.22% less SLUT and 43.48% less FF over the best existing scheme.**

***Index Terms*—Continuous-flow, fast Fourier transform (FFT), memory-based architecture, real-valued signal.**

I. Introduction

Fourier transform plays a very important role in the analysis of signal spectrum by converting a signal from time-domain to frequency-domain, and vice-versa. Discrete Fourier transform (DFT) is the most widely used algorithm. However, its real-time implementation is difficult due to high-computational cost. Cooley and Tukey paved the way for electronic devices by introducing fast Fourier transform (FFT) which computes the DFT efficiently [1]. It is one of the most highly researched algorithm in digital signal processing (DSP), with applications ranging from design of digital filters and orthogonal frequency division multiplexing (OFDM) systems to various consumer electronics such as TV, mobile phones, etc.

Several authors have suggested efficient architectures for computation of complex FFT (CFFT) [2]–[5]. Over the years, the focus has shifted towards efficient implementation of real FFT (RFFT) architectures [6]–[10], as most of the physical signals are real-valued. Moreover, CFFT architectures cannot achieve the same efficiency while processing a RFFT. This is due to Hermitian symmetry in real-valued signal spectrum which eliminates redundant computations. Broadly, FFT architectures are classified as memory-based and pipelined. Memory-based FFT offers low-area and low-power consumption while pipelined FFT offers high-throughput. In the past, there has been continuous efforts to improve the performance of memory-based FFT architectures [8]–[10]. An in-place RFFT processor with low-hardware usage and high-hardware resource utilization has been proposed in [8]. Another architecture has been proposed in [9], to further reduce the hardware usage and the number of computation cycles while maximizing the resource utilization. It is found that the number of cycles required for FFT computation can also be reduced by increasing the number of processing elements (PEs). Further, number of multipliers required for implementation depends on the PEs used, rather than the number of stages. Therefore, hardware cost is relatively lower as compared to pipelined architectures for higher-points. In this paper, we investigate different PEs to reduce the implementation complexity of proposed memory based RFFT processors. The main contributions of this paper are listed as follows:

- Two low-complexity RFFT architectures are presented.
- Memory accessing schemes for continuous-flow operation.

This paper is organized as follows. Section II gives a brief review of existing works. Section III presents the details of the proposed architectures. Section IV compares the performance of the proposed scheme with the recent state-of-the-art. Section V summarizes the main conclusions of the paper.

II. Review of Existing Works

Consider a N-point DFT $X(k)$ of a discrete-time sequence $x(m)$ as

$$X(k) = \sum_{m=0}^{N-1} x(m) W_N^{mk} \tag{1}$$

where $W_N^{mk} = e^{-j(2\pi/N)mk}$ is a twiddle factor for $0 \leq m \leq N-1$ and $0 \leq k \leq N-1$. For real-valued input signals, it is not necessary to compute all the FFT coefficients, since $X(k) = X^*(N-k)$, where $X^*(\cdot)$ indicates conjugate of $X(\cdot)$. Fig. 1 shows data flow graph (DFG) of a 32-point RFFT which computes the outputs in five stages. Few important observation from the DFG are made as follows: The last two stages do not require any twiddle factor multiplication with only one butterfly operation in Stage 5. Also, twiddle factor W^0 is real of value equals to one, therefore, its multiplication can be ignored. Conceptually, memory-based architectures are based on folding technique [11] for stage-by-stage FFT computation.

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00027

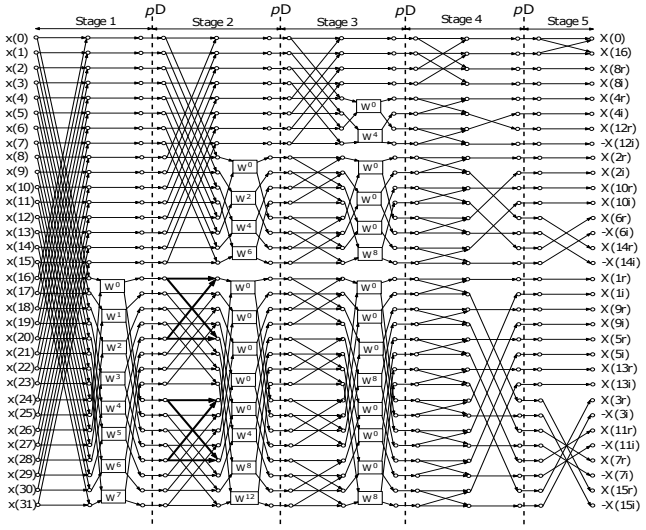


Fig. 1. Data-flow-graph (DFG) of 32-point RFFT [10].

It comprises of one or more PEs and memory units for data-buffering, intermediate computations and output re-ordering. In general, memory-based architectures require $3N$ memories for the computation of N-point FFT. Later, it was established by some researchers that input and output memories can be merged which results in $2N$ memories [2], [3]. This is because the intermediate results of a particular stage have to be written to a memory location from which the next input is to be read and is referred as concurrent input/output (I/O) operation. It is found that the multiplier is one of the main area and power consuming unit in a memory based RFFT processor. In the past, several schemes [2], [8]–[10] have been suggested to reduce the hardware complexity of the multiplier in the PE or the overall complexity of the processor. In case of memory-based RFFT architectures, PE contains fixed number of multipliers and adders. In [8], two butterfly units are present in PE which employs four real adders, four multiplexers, a switch and a complex multiplier. A new stage-partition scheme with a less complex PE was introduced in [9]. However, it suffers from the problems of non continuous-flow and scrambled output data. For an architecture to support continuous-flow, it must perform both computation and data I/O simultaneously. This issue has been solved in [10] by a continuous-flow, normal-order output RFFT processor based on [9]. Although continuous-flow is gained, it leads to increase in the cost of multiplexers and registers. From the above discussion, it is clear that achieving low-hardware complexity and continuous-flow simultaneously is difficult in memory-based RFFT architectures. This motivate us to present two low-complexity RFFT architectures (Type-I, II) that supports continuous-flow and normal-order output. In Type-I, we primarily focus on reducing the multiplexers and registers complexities of PE with proposed memory accessing scheme. In Type-II, we extend the idea of Type-I design to reduce the number of multipliers alongwith a different memory accessing scheme.

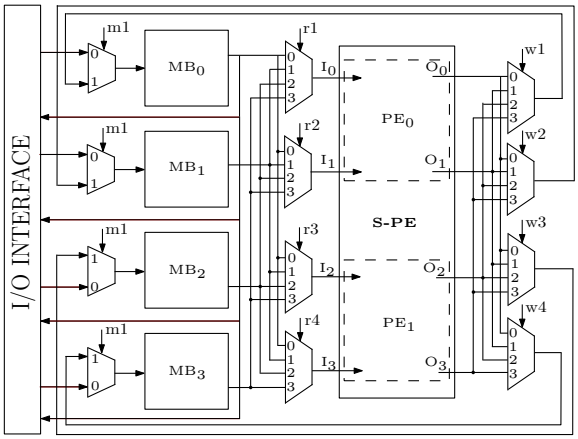


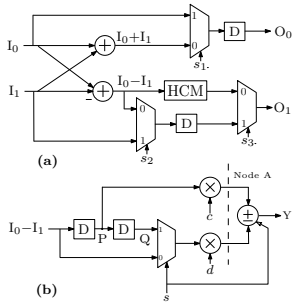
Fig. 2. Top-Level RFFT architecture with one four-input split-PE (Type-I).

III. Proposed Scheme

The proposed RFFT architectures for Type-I, II are based on the DFG presented in [10]. Fig. 1 shows the DFG to compute the butterflies operation and multiplications in the first $(\log_2 N - 2)$ stages sequentially. As earlier stated, last-stage of such DFG requires a single butterfly operation for computation which can be performed in one clock cycle. While the number of clock cycles required for computation from Stage 1 to Stage $(n-1)$ with four-input PE (4-PE) would be $(N/4)(\log_2 N - 1)$, where $n = \log_2 N$. Hence, one can estimate the total number of clock cycles for N-point memory based RFFT computation as $(N/4)(\log_2 N - 1) + 1$. The only difference between the DFG shown in Fig. 1 and existing DFG [10] is that p registers are inserted for pipelining. The value of p decides the number of clock cycles to produce the correct output by the PE in Type-I,II designs. Noticeably, the pipelined registers would increase the number of clock cycles, but reduces the number of multipliers to be employed in a particular PE. For instance, an N-point FFT with one pipelined register in the PE would require additional number of clock cycles over the design reported in [9]. This leads to two real multipliers in each PE instead of one complex multiplier. In such case, the effective number of clock cycles would be $(N/4)(\log_2 N - 0.5) + 1$. In general, if p be the number of pipelined registers to be employed in a four-input PE, then the total number of clock cycles would be $(N/4)(\log_2 N - (1/(p+1))) + 1$. Now, if instead of four inputs, two inputs are used, then total number of clock cycles for N-point RFFT computation would become $(N/2)(\log_2 N - (1/(p+1))) + 1$.

A. Architectural Details of Type-I Design

Fig. 2 shows the top-level architecture of the proposed FFT processor. It consists of four input split-PE (SPE) which has two sub-groups, namely, PE_0 and PE_1. Further, four memory banks MB_0, MB_1, MB_2, MB_3 are used with depth $N/4$ and words size W-bits to store the input data, output data and the intermediate data samples. Thus, the total size of memory required for the implementation of Type-I design is



TIMING DIAGRAM OF HCM

Clk	I_0-I_1	P	Q	s	Node A	Y
0	x_{16}	–	–	–	–	–
1	x_{24}	x_{16}	–	0	cx_{16}, dx_{24}	$cx_{16} - dx_{24}$
2	x_{20}	x_{24}	x_{16}	1	cx_{24}, dx_{16}	$cx_{24} + dx_{16}$
3	x_{28}	x_{20}	x_{24}	0	cx_{20}, dx_{28}	$cx_{20} - dx_{28}$
4	x_{18}	x_{28}	x_{20}	1	cx_{28}, dx_{20}	$cx_{28} + dx_{20}$
5	x_{26}	x_{18}	x_{28}	0	cx_{18}, dx_{26}	$cx_{18} - dx_{26}$
6	x_{22}	x_{26}	x_{18}	1	cx_{26}, dx_{18}	$cx_{26} + dx_{18}$
7	x_{30}	x_{22}	x_{26}	0	cx_{22}, dx_{30}	$cx_{22} - dx_{30}$

(c)

Fig. 3. Type-I design (a) Schematic of $PE_{0,1}$ of a SPE. (b) Schematic of half-complex multiplier (HCM). (c) Timing diagram of HCM.

$4 \times N/4 \times W$. Note that the memory employed have dual ports to perform the read and write operations simultaneously. Initially, all the input samples are stored in respective MB_i $(i = 0, 1, 2, 3)$ via four multiplexers labelled 'm1', as shown in Fig. 2. As mentioned, the computation of every stage requires the intermediate results back to the memory banks. When computation begins, the input samples already stored in the memory banks are read out via another set of multiplexers '$r1, r2, r3$, and $r4$'. It can be noted that the numbers in the multiplexers are in accordance with the memory bank indices 'i'. After processing the samples in the SPE, the four multiplexers '$w1, w2, w3$, and $w4$' decides the write sequence for the intermediate results. In the last two stages, re-ordering and write operation for intermediate computations can be performed concurrently before the next N-point data samples are read into the SPE. The notations of input ports and output ports are I_i and O_i respectively.

In this design, low-complexity SPE and memory addressing scheme are proposed to achieve continuous-flow architecture. Each of the PEs (PE_0 and PE_1) consists of a single butterfly, two registers (D), one multiplexer and a half-complex multiplier (HCM) to process two inputs in parallel, as shown in Fig. 3(a). The control signals to the multiplexers shown in Fig. 3(a) are set to $s_2 = 0$ in the current clock cycle and $s_3 = 1$ in the next clock cycle to by-pass the HCM corresponding to $(n-1)^{\text{th}}$ stage and one butterfly computation in the n^{th} stage, while $s_1 = 1$, $s_2 = 1$ in the current clock cycle and $s_3 = 1$ in the next clock cycle corresponding to n^{th} stage. The schematic of proposed HCM is shown in Fig. 3(b). It consists of two real multipliers, one real adder, one multiplexer and two delay elements. The computation of two butterflies and one twiddle factor multiplication by one PE are performed in two consecutive clock cycles. Thus, the SPE processes eight samples completely in two clock cycles. To understand clearly, an explicit timing diagram for the operation of proposed HCM is shown in Fig. 3(c). The proposed HCM structure leads to less complexity as it involves one multiplexer. Further, registers in HCM are pre-placed to store the samples instead of post-placed in [10] to store the results. By doing so, the register complexity in the proposed HCM is almost reduced by $2W$ which leads to less power consumption over the design for several computation clock cycles.

Stage	Count	$c_2c_1c_0$	$\bar{c}_2c_1c_0$	$c_2\bar{c}_1c_0$	$c_2c_1\bar{c}_0$	$c_2\bar{c}_1\bar{c}_0$	$\bar{c}_2\bar{c}_1\bar{c}_0$
1	0,3,4,7	$MB_{0,2}$	$MB_{1,3}$				
	1,2,5,6					$MB_{1,3}$	$MB_{0,2}$
2	0,2	$MB_{0,3}$		$MB_{1,2}$			
	1,3	$MB_{1,2}$		$MB_{0,3}$			
	4	$MB_{1,3}$			$MB_{0,2}$		
	5			$MB_{1,3}$		$MB_{0,2}$	
	6			$MB_{0,2}$		$MB_{1,3}$	
	7	$MB_{0,2}$			$MB_{1,3}$		
3	0,1	$MB_{0,2}$			$MB_{1,3}$		
	2,3,4,7	$MB_{0,1,2,3}$					
	5,6					$MB_{0,1,2,3}$	
4	0,1	$MB_{0,1,2,3}$					
	2,3	$MB_{1,3}$			$MB_{0,2}$		
	4	$MB_{1,3}$		$MB_{0,2}$			
	5				$MB_{0,2}$	$MB_{1,3}$	
	6				$MB_{1,3}$	$MB_{0,2}$	
	7	$MB_{0,2}$		$MB_{1,3}$			
5	0,3,4,7	$MB_{0,2}$	$MB_{1,3}$				
	1,2,5,6					$MB_{1,3}$	$MB_{0,2}$

(a)

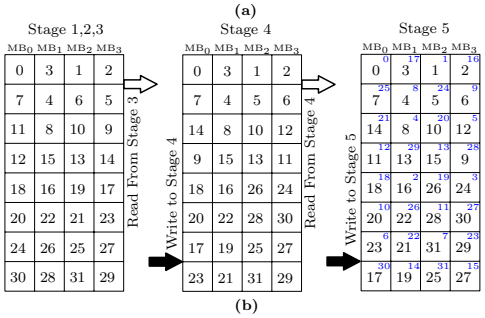


Fig. 4. (a) Address pattern permutations of MB_0, MB_1, MB_2, MB_3 for 32-point RFFT. (b) Memory addressing scheme for 32-point RFFT, where A^b indicates b is time index of final output sample A.

1) Memory Addressing Scheme: In previous discussion, we describe the operation of proposed HCM for both the PE_0 and PE_1 units. In every computation cycle, it is required to read and write the data value in the memory at appropriate address locations. To do that, a conflict-free memory addressing scheme for Type-I RFFT architecture is investigated. In the proposed RFFT, all the input samples are placed in the memory banks from 0 to $N-1$, as shown in Fig. 4(b). It is to be noted that as all four memories are accessed simultaneously, it must be assured that the four samples processed in the PE in every clock cycle come from different memories. For example, in case of a 32-point RFFT, the data with indices $(0, 16, 1, 17)$ are read from their respective memory banks and fed into the PE. Note that all these data belong to different memory banks. The butterflies in the DFG are computed in top-to-bottom order with a number of pre-defined rules. The related butterflies (shown in bold lines in Fig. 1) which enter the same multiplier in the first $(n-2)$ stages are to be computed through the same PE in consecutive cycles. As mentioned earlier, the data in the last two stages require data-reordering to obtain a normal-order output. Hence, in the $(n-1)^{\text{th}}$ stage, the butterflies with the two data samples which interchange their address locations are processed simultaneously. In the n^{th} stage, the data samples that are to be interchanged are read and written in consecutive clock cycles. This leads to continuous-flow operation. For an $N(=2^n)$-point RFFT, a $(n-2)$-bit counter,

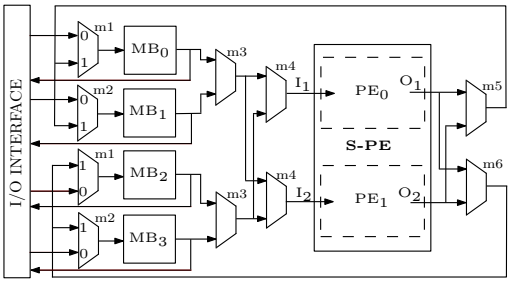
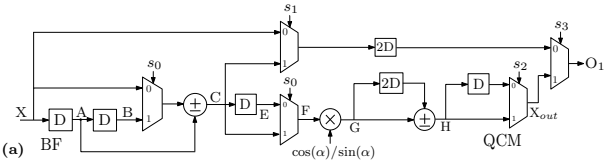


Fig. 5. Top-Level RFFT architecture with one two-input split-PE (Type-II).

$C = c_{n-3}c_{n-4}....c_0$ is used to generate addresses from 0 to $(N/4-1)$ for the four memory banks. The output of counter is physically mapped to address location within each bank across all the stages according to specific patterns. To understand the address patterns and memory addressing scheme clearly, we considered an example of a 32-point RFFT as shown in Fig. 4. A 3-bit counter generates the address locations from 0 to 7 for the four memory banks in each stage. The address of every sample index within a bank changes with each stage according to Fig. 4(a). Each column in Fig. 4(b) corresponds to one memory bank and the numbers correspond to the indices of the input/output samples at each stage. The read and write patterns of $MB_{0,1,2,3}$ for Stage 1, 2 and 3 are same *i.e.* the first $(n-2)$ stages follow in-place computation, while the last two stages need data reordering to obtain a normal order output. Moreover, data is always read from each of the memory banks in top-to-bottom order across all the stages. The last stage requires $N/8$ clock cycles as operations on only $N/4$ samples are carried out. The write and address pattern in the last stage is used to support normal-order output. When the operations in the last stage are over, the final result is tapped out through the output data path. The numbers indicated in blue colour in Fig. 4 indicate the time indices of the samples at the output.

B. Architectural Details of Type-II Design

The top-level architecture of the proposed Type-II RFFT with a two input SPE is shown in Fig. 5. It consists of four memory banks, a SPE, an I/O interface and sets of multiplexers. The memory banks, MB_0, MB_1, MB_2, MB_3 work in groups of two at every stage. These two groups switches between two operational modes: computational and concurrent I/O. In other words, two memory banks are always working in the computational mode while the remaining two are working in the concurrent I/O mode. The multiplexers, m1 and m2 facilitates the memory groups in switching between the I/O interface and the intermediate results. The multiplexers, m3 and m4 select the two memory banks whose data are to be read simultaneously into the SPE. Finally, m5 and m6 selects the banks to which the processed output is written. The control of these multiplexers alternate for consecutive symbols. In this design, SPE consists of two single-input PEs, PE_0 and PE_1. The architectural details of the single-input $PE_{0,1}$ is shown in Fig. 6(a). Unlike Type-I design, it has a half-butterfly unit and



(a)

TIMING DIAGRAM OF PROPOSED PE IN STAGE 1

Clk	X	A	B	s_0	C	s_1	E	F	G	H	s_2	s_3	O_1
0	x_0	–	–	–	–	–	–	–	–	–	–	–	–
1	x_{16}	x_0	–	0	x_0^1	1	–	–	–	–	–	–	–
2	x_8	x_{16}	x_0	1	x_{16}^1	–	–	x_{16}^1	cx_{16}^1	–	–	–	–
3	x_{24}	x_8	x_{16}	0	x_8^1	1	x_{16}^1	x_{16}^1	dx_{16}^1	–	–	0	x_0^1
4	x_4	x_{24}	x_8	1	x_{24}^1	–	–	x_{24}^1	dx_{24}^1	$cx_{16}^1 + dx_{24}^1$	1	1	$cx_{16}^1 + dx_{24}^1$
5	x_{20}	x_4	x_{24}	0	x_4^1	1	x_{24}^1	x_{24}^1	cx_{24}^1	$cx_{24}^1 - dx_{16}^1$	–	0	x_8^1
6	x_{12}	x_{20}	x_4	1	x_{20}^1	–	–	x_{20}^1	cx_{20}^1	–	0	1	$cx_{24}^1 - dx_{16}^1$
7	x_{28}	x_{12}	x_{20}	0	x_{12}^1	1	x_{20}^1	x_{20}^1	dx_{20}^1	–	–	0	x_4^1
8	x_2	x_{28}	x_{12}	1	x_{28}^1	–	–	x_{28}^1	dx_{28}^1	$cx_{20}^1 - dx_{28}^1$	1	1	$cx_{20}^1 - dx_{28}^1$
9	x_{18}	x_2	x_{28}	0	x_2^1	1	x_{28}^1	x_{28}^1	cx_{28}^1	$cx_{28}^1 + dx_{20}^1$	–	0	x_{12}^1
10	x_{10}	x_{18}	x_2	1	x_{18}^1	–	–	x_{18}^1	cx_{18}^1	–	0	1	$cx_{28}^1 + dx_{20}^1$
11	x_{26}	x_{10}	x_{18}	0	x_{10}^1	1	x_{18}^1	x_{18}^1	dx_{18}^1	–	–	0	x_2^1
12	x_6	x_{26}	x_{10}	1	x_{26}^1	–	–	x_{26}^1	dx_{26}^1	$cx_{18}^1 - dx_{26}^1$	1	1	$cx_{18}^1 - dx_{26}^1$

(b)

Fig. 6. Type-II design (a) Schematic of $PE_{0,1}$ of a SPE. (b) Schematic of quarter-complex multiplier (QCM). (c) Timing diagram of $PE_{0,1}$ including QCM operation, where x_m^n is output of n^{th} stage with sample index m.

Algorithm 1 Compute N-point RFFT with i-input PE

Notations:
WP: write port, RP: read port, Addr: MB_i address
Initialize:
$C_i = 0 \ \forall \ l \in [0, N/i-1]; m, k \in [0, N-1]$
loop
$X(k) = \sum_{m=0}^{N-1} x(m) W_N^{mk}$
  if $E_{\overline{WP}/RP} == 0$ **then**
    $I_i[\text{MB}_i^{C_i'}[O_i]] \leftarrow O_i$ // WP[$\text{MB}_i^{\text{Addr}}$[RP]]
  else
    $I_i \leftarrow I_i[\text{MB}_i^{C_i'}[O_i]]$
  end if
  for $n = 0$ to $\log_2 N - 1$ **do**
    for $l = 0$ to $N/i - 1$ **do**
      $C = C + 1$
      load $C_i = C$
      map $C_i' = C_i$ // (refer Fig. 4(a))
    end for
  end for
end loop

a new quarter-complex multiplier (QCM) unit to process the data from a memory bank. To clearly understand the operation of QCM unit, it is convenient to include the half-BF unit with QCM, as shown in Fig. 6(a). Both half-BF and QCM units operate sequentially during the first $(n-2)$ stages on sample-by-sample basis in the following two modes:

- It performs the addition using the half-BF unit in one clock cycle and the result is passed via the by-pass line with two-delay units to the output multiplexer.
- It performs the subtraction using the half-BF unit in the next clock cycle and the result is processed by the

TABLE I
COMPUTATIONAL COMPLEXITIES OF VARIOUS MEMORY-BASED FFT

Design	MULT	ADD	REG	MEM	MUX/ DEMUX	CC	CP
Jo et al. [2]	12	22	0	$4N$	38	$(N/4)\log_4 N$	$T_{MA}+2T_A+8T_{MX}+2T_M$
Ayinala et al. [8]	4	6	0	$2N$	38	$(N/4)\log_2 N$	$T_{MA}+T_A+8T_{MX}+2T_M$
Ma et al. [9]	4	6	0	$2N$	28	$(N/4)(\log_2 N-1)+1$	$T_{MA}+T_A+5T_{MX}+2T_M$
Mao et al. [10]	4	6	10	$2N$	50	$(N/4)(\log_2 N-(1/2))+1$	$T_{MA}+T_A+11T_{MX}+2T_M$
Proposed Type-I	4	6	8	$2N$	36	$(N/4)(\log_2 N-(1/2))+1$	$T_{MA}+T_A+6T_{MX}+2T_M$
Proposed Type-II	2	4	16	$2N$	20	$(N/2)(\log_2 N-(1/3))+2$	$T_{MA}+T_A+7T_{MX}+2T_M$

MULT: Multipliers, ADD: Adders, REG: Registers, MEM: Memory, MUX/DEMUX: Multiplexers/De-multiplexers, CC: clock cycles, CP: clock period, Throughput = 1/(CC×CP), T_{MA}, T_A, T_{MX} and T_M are delays of multiply-add unit, 16-bit adder, 2-to-1 multiplexer and memory-access, respectively.

QCM. Subsequently, it undergoes real multiplication in the QCM unit which takes two clock cycles to compute $cx_j + dx_k$ (or $cx_k - dx_j$) using the four quarter complex terms cx_j, cx_k, $+dx_k$ and $-dx_j$, where j, k denote the sample indices with $j \neq k$, and c (or $\cos(\alpha)$) and d (or $\sin(\alpha)$) are real and imaginary components of twiddle-factor respectively.

The proposed QCM unit comprises of one real adder, one real multiplier, two multiplexers and four registers. The complete operation of proposed QCM is explained using a separate timing diagram with half-BF unit in Stage 1, as shown in Fig. 6(b). Note that the letters in the table correspond to various intermediate nodes of half-BF and the QCM units.

1) Memory Addressing Scheme: The memory addressing scheme and address patterns for Type-II design is same as that of Type-I design. The only difference lies in the way of accessing the memory banks. For instance, in Type-I, the SPE accesses all the four banks simultaneously while in the case of Type-II, the SPE accesses only two of the banks at a time. Note that in Type-II design, these two memory banks form one group. In Stage 1 and Stage 3, (MB_0, MB_2) forms one group while (MB_1, MB_3) forms the second group. In Stage 2, for the upper half, the two groups are (MB_0, MB_3) and (MB_1, MB_2) while for the lower half, the two groups are (MB_0, MB_2) and (MB_1, MB_3). In Stage 4, for $C = 0$, (MB_0, MB_2), (MB_1, MB_3); for $C = 1$, (MB_0, MB_3), (MB_1, MB_2) and for remaining, (MB_0, MB_1), (MB_2, MB_3). In Stage 5, groups are (MB_0, MB_2), (MB_1, MB_3). The butterflies in the DFG are computed in top-bottom order with the pre-defined rules of Type-I, as discussed earlier. The read and write patterns are in accordance with the DFG. The write cycle is three copy delay of the read cycle for the first $(n-2)$ stages as the PE has a latency of three. In Stage $(n-1)$ and Stage n, the write cycle changes the memory banks of the processed samples that require data re-ordering according to the DFG shown in Fig. 1. Algorithm 1 explains the operation of Type-I,II designs.

IV. PERFORMANCE COMPARISON

In this section, we first compare the hardware and time complexities of the proposed and existing designs. Next, we evaluate and compare the performance of Type-I, II architectures along with the existing designs through ASIC synthesis and FPGA implementation in terms of area, power, minimum-clock period (MCP), power-delay product (PDP), slice look-up tables (SLUT) and flip-flops (FF).

A. Hardware and Time Complexities

The estimated hardware and time complexities of the proposed and existing schemes in terms of mutlipliers, adders, multiplexers, memory and throughput complexities are listed in Table I. The architecture in [2] supports continuous-flow operation, but involves redundant operations when applied to RFFT computations. In the sequel, RFFT architectures reduce the memory by a factor of two over CFFT architectures. To reduce the hardware computations, we present two new designs for RFFT computations. The existing architectures [8], [9] neither supports concurrent I/O operation nor support continuous-flow operation. However, the design presented in [10] supports continuous-flow operation, but requires high complexity of registers and more number of multiplexers. To address this issue, we moved the locations of multipliers towards the output side, while the inputs to the HCM are directly stored in the registers in the Type-I design. The complexity of multiplier is further decreased in Type-II design by employing a QCM. When these computational units are used over several clock cycles, it leads to increase in power consumption. Since multiplication always result in increase of wordlength to $2W$ when the operands have wordlength of W-bits. Therefore, the PE of Type-I design allows reduction in power consumption. The proposed Type-I requires $2W$ less registers per clock cycle compared to [10], therefore total utilization of registers for $2W \times [(N/4)(\log_2 N - 0.5) + 1]$. The savings have direct impact on dynamic power consumption and it would be higher for both memory wordlengths W. Further, Type-I design uses less number of multiplexers both in PE and memory access, while using the same number of multipliers and adders as that of [8], [9] and [10]. It is important to note that the number of multiplexers in Type-II design is least among all the existing schemes while Type-I uses less number of registers over [10].

We have also listed the time-complexities of different designs in terms of throughput. It is determined by the clock-period (CP) (or critical path) and number of clock cycles (CC) involved of the design, as listed in Table II. As shown in Table II, the critical path of proposed Type-I design is reduced as compared to [10], while it is same as that of [8]. On the other hand, number of clock cycles required for the proposed design is less always than [8] which leads to higher throughput.

TABLE II
SYNTHESIS RESULTS ON TSMC 90NM CMOS AND XILINX FPGA

Design	N	Area (μm^2)	Power (mW)	MCP (ns)	PDP (mW.ns)	SLUT	FF
Jo et al. [2]	16	88124	10.92	5.13	56.01	3291	3487
	32	169231	22.03	6.18	136.14	6253	6501
Ayinala et al. [8]	16	31921	4.62	2.14	9.89	1198	1215
	32	43676	7.13	2.17	15.47	2435	2502
Ma et al. [9]	16	32253	4.71	2.25	10.59	1312	1347
	32	45238	7.42	2.33	17.29	2652	2763
Mao et al. [10]	16	35497	4.97	2.42	12.03	1485	1507
	32	49187	7.81	2.51	19.60	2853	2932
Proposed Type-I	16	34078	4.22	2.31	9.75	1394	1432
	32	46698	6.42	2.38	16.47	2665	2778
Proposed Type-II	16	18578	2.88	4.42	10.83	783	881
	32	24903	4.40	4.49	16.12	1477	1657

While the scheme in [9] consumes second least number of clock cycles and clock period, however it uses non-continuous flow. Furthermore, the number of clock cycles required in the proposed scheme is same as that of the architecture in [10]. However, the computation time required to compute the N-point RFFT is lesser than [10] due to shorter clock period. In above definition, [9] is considered as the reference due to aforesaid reasons.

B. ASIC and FPGA Implementation Results

The proposed and existing designs are coded in Verilog for 16 and 32-points FFT. Application specific integrated circuit (ASIC) synthesis is performed by Cadence 14.1 RTL compiler using TSMC 90 nm CMOS technology. The corresponding synthesis results of Type-I,II designs in terms of area, power, MCP, and PDP are listed in Table II. It can be noted that the proposed Type-I design occupies slightly lesser area, while it consumes significantly lesser power as compared to existing designs. Further, there is also marginal reduction in MCP for the Type-I design since less combinational logic delay is involved for the RFFT computation over the design [10]. For example, a 32-point Type-I RFFT occupies nearly 5.06% less area, 15.1% less power, 6.58% less SLUT and 5.25% less FF while Type-II offers 47.76% less area, 43.64% power, 48.22% less SLUT and 43.48% less FF over the RFFT in [10]. In case of proposed Type-II design, the amount of area and power reduction are substantial since multipliers complexity are significantly reduced as compared to Type-I design. This comes at the cost of increased MCP over the Type-I design. For better comparison, we have estimated PDP values of different designs for their computational cycles, and corresponding results are listed in Table II.

The proposed and existing designs are also implemented on a Xilinx ZYNQ FPGA device (XC7Z020-1CLG84C) for 16-point and 32-point RFFT. The logic utilization is obtained in terms of slice LUTs (SLUT) and flip-flops (FF) by setting the system clock at 50 MHz, as listed in Table II. From the implementation results, it is clear that the proposed Type-I design for 16-point RFFT has almost 5.45% less SLUT and 4.97% less FF over best existing design [10]. On the other hand, the proposed Type-II design offers 45.66% less SLUT and 41.86% less over the design in [10].

V. CONCLUSION

This paper presents two new memory-based architectures for RFFT computation based on novel conflict-free memory access strategies to achieve continuous-flow and a normal-order output. In Type-I design, a new approach has been developed to minimize the multiplexers and registers complexity. Furthermore, for the same number of clock cycles, the proposed scheme requires lesser computation time to compute the N-point RFFT than [10] due to shorter clock period. This makes it possible to achieve a reduction in power consumption and increase the throughput. The proposed RFFT has been implemented efficiently on an FPGA making use of logic elements. Further reductions can be achieved for higher-point FFT computations. The proposed PE achieves lower hardware usage and complexity as compared to the recent schemes. Synthesis results validates the feasibility of the proposed designs. The most important feature of proposed scheme is that the architecture is continuous with low-hardware complexity and is applicable to real-time processing.

VI. ACKNOWLEDGEMENT

This work was supported by Special Manpower Development Programme for Chip to System Design (SMDP-C2SD) sponsored by the Ministry of Electronics & Information Technology (MeitY), Govt. of India.

REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[2] B. G. Jo and M. H. Sunwoo, "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 5, pp. 911–919, 2005.

[3] P.-Y. Tsai and C.-Y. Lin, "A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 12, pp. 2290–2302, 2011.

[4] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1068–1081, 2012.

[5] J. Liu, Q. Xing, X. Yin, X. Mao, and F. Yu, "Pipelined architecture for a radix-2 fast walsh–hadamard–fourier transform algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 11, pp. 1083–1087, 2015.

[6] H. V. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 6, pp. 849–863, 1987.

[7] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 12, pp. 2634–2643, 2009.

[8] M. Ayinala, Y. Lao, and K. K. Parhi, "An in-place FFT architecture for real-valued signals," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 10, pp. 652–656, 2013.

[9] Z.-G. Ma, X.-B. Yin, and F. Yu, "A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 9, pp. 876–880, 2015.

[10] X. Mao, Z. Ma, F. Yu, and Q. Xing, "A continuous-flow memory-based architecture for real-valued FFT," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 11, pp. 1352–1356, 2017.

[11] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*. John Wiley & Sons, 2007.