International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 24147-24164
© Research India Publications
http://www.ripublication.com

Implementation of 4096 Point FFT Using Modified 4 Point Radix 2 FFT Kernels To Reduce Latency

Dr. Amos Jeeva Oli H
Head, Department of Electronics and Communication Engineering,
K. C. G. College of Technology, Karapakkam. Chennai 600097
[email:amosjeeva@gmail.com]

Abstract

An OFDM transceiver suffers from latency due to the calculation of FFT/IFFT. The objective of this paper is to exemplify a new approach for performing a 4k FFT in order to reduce latency by operating on the inputs as they arrive. The large sized FFT is realized using smaller FFT kernels as it does not increase the computational complexity. 4 point radix-2 FFT/IFFT is used since only adders are needed to develop the kernel which is used to evaluate 4096 point FFT. The large sized FFT is calculated by using these kernels over six pipeline stages and in parallel within stages. This approach significantly reduces the latency in the OFDM transceiver and usage of a modified 4 point radix 2 FFT/IFFT as the basic unit to evaluate the large 4096 point FFT/IFFT also reduces the computational requirements. The same has been implemented in a Spartan 3 FPGA and the latency is significantly reduced.

Keywords: latency, FFT, IFFT, OFDM,

Introduction

Orthogonal Frequency division multiplexing which is used in the fourth generation mobile communication systems uses Inverse Fast Fourier Transform and Fast Fourier Transform functionalities in its endeavor to multiplex closely spaced carriers which carry low rate data to mitigate all ill-effects that arise due to multipath configuration. The devices that use 4G are mostly mobile and handheld, i.e., standalone and compact, hence IFFT/FFT blocks which are computationally less complex and consume less power are preferred. These devices support several real time applications hence reduced latency is necessary. In this paper large sized IFFT/FFTs are decomposed into smaller FFTs using 'divide and conquer approach', in order to reduce latency. In order to reduce time consumed, larger FFTs are decomposed into smaller FFTs, and 'less time consuming' additions are used for the computation of these FFTs. The evaluation of the IFFT/FFT is initiated as the inputs arrive and hence the need to wait for the arrival of all the inputs does not arise.

A 4k point FFT is implemented on a Xilinx FPGA using six stages, the execution of every stage does not wait for the complete execution of the preceding stage; rather it starts execution on the available outputs from the preceding stage. The objective is to act upon the inputs as they arrive and not to wait for the arrival of set of inputs for the commencement of the IFFT/FFT function. The number of inputs on which the IFFT/FFT function is to be performed is 4096 precisely. The conventional approach is to perform IFFT/FFT after all the 4096 symbols arrive and the latency will be greater than the sum total of the symbol period of all these symbols in any OFDM system. The latency in this work is just a few symbol periods after the arrival of the last symbol.

Related Work

Adiono, T. et al [1] presented an implementation of a parallel-pipelined configurable FFT/IFFT processor for Orthogonal Frequency Division Multiple Access (OFDMA) applications in LTE. The architecture combines a multipath delay commutator and single-path delay feedback style to obtain low latency, high throughput, and high efficiency memory. A radix 2^2 SDF pipelined FFT/IFFT processor was implemented with an architecture that had the same multiplicative complexity as radix-4 but retains the butterfly structure of radix-2 by Ahmed Saeed et al [2]. The results showed that the processor achieves higher throughput and lower area and latency. Alexander A Petrovsky et al [3] created a methodology for automatic synthesis of real time FFT processors at structural level under the given restrictions: speed of input data receipt, structure of the computing element and the time of the butterfly execution. It involved creating parallel-pipelined structures for fixed radix FFT and modified split radix FFT algorithms. A VLSI FFT architecture based on combining three consecutive radix-4 stages to result in a 64-point FFT engine was presented by Babionitakis K. et al [4] Cascading these 64 point FFT engines resulted in an improved architecture design with reduced memory requirements and latency reduced to one third compared to the fully unfolded radix-4 architecture. Jesus Garcia et al [5] have used architecture for FPGA implementation of a Split-Radix FFT processor, which combines the higher parallelism of the 4r-FFTs and the possibility of processing sequences having length of any power of two. The simultaneous operation of the multipliers and adder-subtractors implicit in Split-radix FFT lead to faster operation. Bin Zhou et al [6] presented optimized implementations of two different pipeline FFT processo rs. Different optimization techniques and rounding schemes were explored. The implementation of 16 bit 1024 point FFT with the R2^2SDF architecture achieved better performance with lower resource usage than prior art. The R2^2SDF was more efficient than the R4SDC in terms of throughput per area due to a simpler controller and an easier balanced rounding scheme. They also showed that balanced stage rounding is an appropriate rounding scheme for pipeline FFT processors. A reconfigurable FFT architecture which processes variable-length, multi-streams, namely, 1 stream of 2048 point FFT or 2 streams of 1024 point FFT or 4 streams of

512 point FFT using modified radix-2 single delay feedback (SDF) FFT was introduced by Boopal P.P et al [7]. This architecture achieves the throughput that is required by the WiMax standard. C. Yu et al [8] implemented a pipeline FFT/IFFT processor adopting a single path delay feedback by using a reconfigurable complex multiplier and bit-parallel multipliers to achieve a ROM-less FFT/IFFT processor. In order to reduce the latency of the FFT block, the FFT is modified to allow calculation of FFT as the input arrives. The individual input's contribution towards outputs are identified in the form of equations for radix-2 FFT algorithm, as its regular structure is attractive for high throughput design as stated by Chao Cheng et al [9]. A 256 point FFT architecture that utilizes cascaded radix-2^4 single-path delay feedback (SDF) structures based on radix-16 FFT algorithm with low multiplier and multiplication complexities and a simple control circuitry was proposed by Chih-Peng Fan et al [10]. Chin-Long Wey et al [11] presented simple radix-2 memory-based FFT (MBFFT) processors with low hardware cost and high maximum operation frequency. Chin-Long Wey et al [12] also presented parallel MBFFT structures with two and four butterfly processing elements (PEs) respectively, to improve the latency while still keeping hardware cost low and maximum operation frequency high. An multiplier less VLSI architectures of Split radix FFT algorithm using new distributed arithmetic (NEDA) was introduced by DiptiSankar Das et al [13]. Since the architecture did not contain any multiplier blocks, reduction in terms of power, speed and area was observed. A fixed-point, 16-bit word-width, 64-point FFT/IFFT processor was developed primarily for the application in an OFDM-based IEEE 802.11a wireless LAN baseband processor by Koushik Maharatna et al [14]. The 64-point FFT was realized by decomposing it into a two-dimensional structure of 8-point FFTs. This approach reduced the number of required complex multiplications compared to the conventional radix-2 64-point FFT algorithm. The complex multiplication operations were realized using shift-and-add operations. Thus, the processor did not use a two-input digital multiplier. It also did not need any RAM or ROM for internal storage of coefficients. A modified single-path delay feedback (SDF) architecture for FFT implementation, which implements a mixed decimation-in-frequency (DIF) / decimation-in-time (DIT) FFT algorithm was proposed by Seungbeom Lee et al [15]. This architecture was applied to a 64-point FFT and compared to the radix-4 DIF SDF and radix-4 multi-path delay commutator (MDC) architecture in the context of throughput, latency and hardware complexity. It exhibited lower hardware complexity as compared to the radix-4 MDC while maintaining the same throughput and latency. It achieved lower latency compared to the original radix-4 SDF architecture with reasonable increase in hardware complexity. Xu Peng et al [16], chose split-radix algorithm as the basic algorithm and implemented High speed FFT algorithm using paralleled processing and pipeline techniques. This method performed well when implemented in FPGA and satisfied the requirement of high speed. Results show the system latency of 13 clock periods and high efficiency in conserving hardware resources.

Modification of 4 point radix-2 FFT/IFFT

Divide and conquer approach

Generally a N point DFT is described as below, where x(n) has N time samples and it results in N frequency samples.

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn} \; 0 \le k \le N-1$$

Alternatively, N point IDFT helps obtain N time samples from N frequency samples as given below,

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk} \; 0 \le n \le N-1$$

Both the expressions use the twiddle factor

$$W_N^{nk} = e^{-j2\pi/N}$$

which exhibits the following properties.

$$W_N^{k+N/2} = -W_N^k$$
$$W_N^{k+N} = W_N^k$$

Consider a discrete time sequence of length N>>1, N can be factored as LM. This N point DFT can be performed by arranging N points in L rows and M columns.

$$N = LM, L - rows, M - columns$$
$$0 \le l \le L-1$$
$$0 \le m \le M-1$$

Similarly the resulting $X(k), 0 \le k \le N-1$ will also appear in a similar arrangement.

$$N = LM, L - rows, M - columns$$
$$0 \le p \le L-1$$
$$0 \le q \le M-1$$

There is a two pronged approach used for converting a single dimensional arrangement into a two dimensional arrangement to represent the DT signal. The sequence x(n) could be arranged by filling row after row. The mapping of n to (row 'l', column 'm') is given by $n = Ml + m$. N point DFT can be performed by first performing L point DFT column-wise to form F (p, m), then multiplying the matrix values with W_n^{pm} and finally performing M point DFT row-wise. The resulting X (k) will appear column-by-column in the proper order, where $k = p + qL$.

In case the input x(n) is mapped column-by-column $n = l + mL$, N point DFT can be performed by performing M point DFT row-wise to give F(l,q), then multiplying the matrix by W_n^{lq} and finally performing L point DFT column-wise. The resulting X(k) will appear row-by-row in the proper order, where $k = Mp + q$.

The latter approach explained employs Column wise mapping for x(n) and row wise mapping for X(k). The DFT equation in such a case is given by

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{i=0}^{L-1} x(l,m) W_N^{(Mp+q)(mL+l)}$$

Rewriting the above equation by expanding the twiddle factors and rearranging the summations results in

$$X(p,q) = \sum_{i=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} x(l,m) W_M^{mq} \right] \right\} W_L^{lp}$$

Decomposing the above equation we obtain the following equation which represents M point DFT row by row.

$$F(l,q) = \sum_{m=0}^{M-1} x(l,m) W_M^{mq} \; 0 \le q \le M-1$$

The matrix then is multiplied by the twiddle factor, every element in the matrix is multiplied by w_n^{lq}, where $0 \le l \le L-1$ and $0 \le q \le M-1$

$$G(l,q) = w_n^{lq} F(l,q)$$

Finally L point DFT is performed column by column and the resulting X(k) is available row wise.

$$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp} \; 0 \le p \le L-1$$

A 4096 point DFT is performed by the above mentioned approach, that is, instead of performing a 4096 point DFT directly, it is rearranged as 4 x 1024, implying that four 1024 point DFTs, twiddle factor multiplication with 4096 elements of the array and 1024 4-point DFTs are required. Instead of performing a 1024 point DFT directly it can be further decomposed into 4 x 256. Every 1024 point DFT will involve 4 256-point DFTs, twiddle factor multiplication with 1024 elements of the matrix and 256 4-point DFTs. This decomposition can continue till it reaches a point that the DFTs that are to be found out is just 4 point DFTs but the evaluation of the 4 point FFTs is on the basis of ‘part-by-part-evaluation-on-arrival’ (PPEA) approach.

Modified radix-2 IFFT/FFT for N=4

Consider 4 time samples $a_r + a_i, b_r + b_i,\ c_r + c_i,\ d_r + d_i$ and frequency samples $A_r + A_i,\ B_r + B_i,\ C_r + C_i,\ D_r + D_i$ and on performing 4-point FFT on the former the latter is obtained. And on performing 4 point IFFT on the latter the reverse is obtained. If the relationship between the inputs and outputs when 4 point radix 2

IFFT/FFT is performed and analyzed, it is found that evaluation is simpler, since no multiplication is involved. It is presented below for analysis and observation.

IFFT

n=0; $a_r = A_r + B_r + C_r + D_r : a_i = A_i + B_i + C_i + D_i$

n=1; $b_r = A_r - B_i - C_r + D_i : b_i = A_i + B_r - C_i - D_r$

n=2; $c_r = A_r - B_r + C_r - D_r : c_i = A_i - B_i + C_i - D_i$

n=3; $d_r = A_r + B_i - C_r - D_i : d_i = A_i - B_r - C_i + D_r$

The relationship between inputs and outputs when 4 point radix 2 FFT is performed also requires only negated inputs for calculating outputs, as is shown below.

FFT

n=0; $A_r = a_r + b_r + c_r + d_r : A_i = a_i + b_i + c_i + d_i$

n=1; $B_r = a_r + b_i - c_r - d_i : B_i = a_i - b_r - c_i + d_r$

n=2; $C_r = a_r - b_r + c_r - d_r : C_i = a_i - b_i + c_i - d_i$

n=3; $D_r = a_r - b_i - c_r + d_i : D_i = a_i + b_r - c_i - d_r$

The PPEA FFT for 4 points, which was developed using radix 2 FFT did not involve any twiddle factor multiplications. 4 point PPEA FFT involves only complex additions or subtractions. When the same is used for realizing a larger sized FFT the resulting operations need complex multipliers for twiddle factor multiplications. The FFT is calculated using 4 point PPEA FFT – which does not involve any complex multiplication - by using pipelined stages. These stages are pipelined and overlapped, implying that the inputs to the consecutive stages are not the complete set of outputs from the respective previous stage but individual outputs are fed as inputs to the consecutive stages. The reduction in latency that is obtained in small sized FFTs is exploited and accrued by this overlapping to result in increased latency. The inputs and outputs that are obtained are not reordered by using this approach.

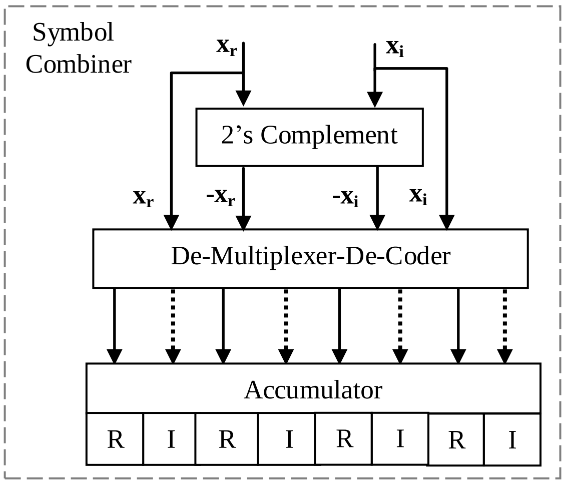


Figure 1: Symbol Combiner

Every consecutive 1024 inputs as they arrive are operated upon by the symbol combiner, shown in Fig. 1 to produce eight outputs to be stored in the 4 real and 4 imaginary storage locations that are 1024 locations apart. For example 0th input will cause outputs to be stored in 0th, 1024th, 2048th and 3072nd locations. These outputs will be used again to be added to the symbol combiner outputs for the 1024th input, the same will be done when the 2048th and 3072nd inputs arrive and are sent to the symbol combiner. It should be noted that to obtain the 4 point FFT outputs the symbol combiner outputs of 0th, 1024th, 2048th and 3072nd inputs are accumulated. Hence for consecutive inputs, results of the symbol combiner are stored and retrieved for accumulation. The symbol combiner operates on the 0-1023 inputs in a similar fashion while it treats the consecutive 3 sets of 1024 inputs in 3 different ways. The net result of what is done is that the inputs have been reordered and 1024 4-point FFTs of the reordered sequences has been performed. Every stage performs a 4 point FFT and twiddle factor multiplication. The subsequent stages need to multiply the twiddle factors and perform 4-point DFTs column-wise. Once again a symbol combiner is preferred to perform the column wise FFTs since the inputs can arrive at different instants of time.

4096 POINT FFT – PPEA APPROACH

Stage wise PPEA FFT

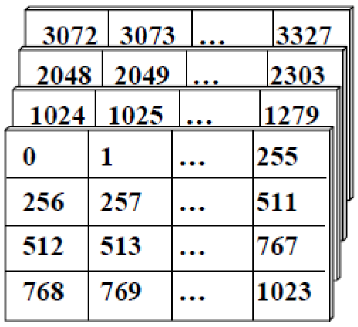
The 4096 point FFT using "divide and conquer" approach is realized by arranging the sequence values row-wise in a matrix of dimension 4x1024 as shown in Fig. 2. In order to obtain a 4096 point FFT, one thousand and twenty four column-wise 4 point FFT is performed, followed by twiddle factor multiplication of every resulting value by W_{4096}^{r1c1}, where r1 = 0 to 3 and c1 = 0 to 1023 and finally four 1024 point FFTs is found. Instead of performing the 1024 point FFTs, they are in turn factorized into a 4 x 256 matrix arrangement as shown in Fig. 2. Hence every 1024 point FFT is performed using two hundred and fifty six column-wise 4 point FFT, followed by multiplication of twiddle factors W_{1024}^{r2c2}, where r2 = 0 to 3 and c2 = 0 to 255 and finally four row-wise 256 point FFT.

Similarly every 256 point FFT is performed by sixty four column wise 4 point FFT in a 4 x 64 matrix arrangement as shown in Figure 5.6, followed by multiplication of twiddle factors W_{256}^{r3c3}, where r3 = 0 to 3 and c3 = 0 to 63 and four row-wise 64 point FFT. Further every 64 point FFT is performed by sixteen column wise 4 point FFT in a 4 x 16 matrix as shown in Figure 5.7, followed by multiplication of twiddle factor W_{64}^{r4c4}, where r4 = 0 to 3 and c4 = 0 to 15 and four 16 point FFTs. Finally every 16 point FFT is performed using 4 column wise 4 point FFT followed by multiplication of twiddle factors W_{16}^{r5c5}, where r5 = 0 to 3 and c5 = 0 to 3 and 4 row wise 4 point FFT in a 4 x 4 matrix arrangement shown in Figure 5.8. In the discussion above every instance where a 4 point FFT is mentioned, a PPEA FFT is used. The total calculation is organized in 6 stages, where the first 5 stages consist of 2 sub-stages, namely PPEA FFT and TWF multiplication while the last stage only consists of only the PPEA FFT.

0	1	…	1023
1024	1025	…	2047
2048	2049	…	3071
3072	3073	…	4095

Figure 2: Stage 1 arrangement of PPEA FFT values, one 4 x 1024 matrix of points

In Stage 1 every input as it arrives undergoes 4 point PPEA FFT between points that are 1024 inputs apart. For the first 1024 inputs, 1024 stage 1 PPEA FFTs are initiated. Thereafter for every subsequent 1024 inputs are the second symbols of stage 1 PPEA FFTs. The next 1024 inputs are the third input to the same, While the 3072nd input arrives, the first of 1024 stage 1 PPEA FFT is completed. Four outputs of the stage 1 PPEA FFT is available for stage 1 Twiddle factor multiplication by a factor,W_{4096}^{r1c1} where r1 = 0, 1, 2, 3 & c1 increments when the successive inputs arrive. Since the twiddle factor is unity when r1=0, multipliers are not used, hence only 3 complex multipliers are needed.



3072	3073	…	3327

2048	2049	…	2303

1024	1025	…	1279

0	1	…	255
256	257	…	511
512	513	…	767
768	769	…	1023

Figure 3: Stage 2 arrangement of PPEA FFT values, Four 4x256 matrix of points

The outputs of the complex multipliers initiate four stage 2 PPEA FFT. This stage 2 FFT involves values that are 256 point apart as shown in Fig. 3. Hence for the next 256 clock instants when consecutive inputs arrive stage 2 PPEA FFTs are initiated, while every consecutive set of 256 clock instants the consecutive inputs for the stage 2 FFT will generated and acted upon by the symbol combiner. When the 3840th input arrives, the stage 1 PPEA FFT involving that input is completed and subsequently 3 stage 1 TWF multiplications are completed, thereafter completing the first four of the stage 2 PPEA FFT. These four stage 2 PPEA FFT involves values that resulted from the completed stage 1 PPEA FFT using inputs (768, 1792, 2816, 3840). Since four stage 2 PPEA FFT is completed, 16 values are available for stage 2 TWF multiplication by a factor $W_{1024}^{r2c2}$where r2= 0 to 3 and c2 = 0 to 255. Complex multipliers use 4 distinct twiddle factors to perform the sixteen stage 2 TWF multiplication. Once again one of the distinct twiddle factor values is unity (r = 0) hence 12 complex multipliers are used.

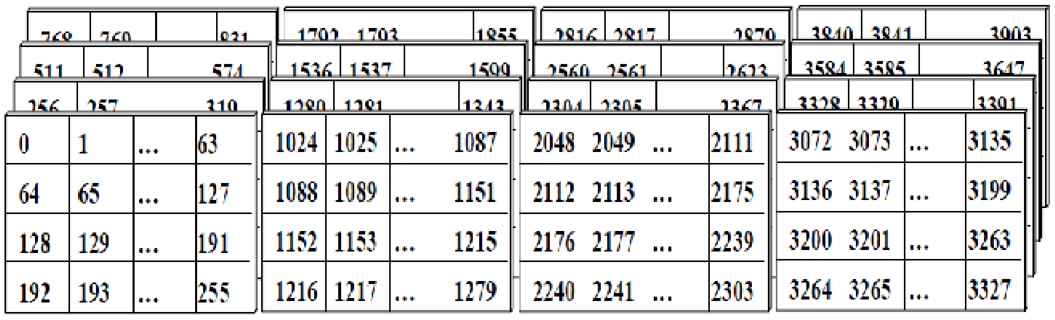


Figure 4: Stage 3 arrangement of PPEA FFT values, sixteen 4x64 matrix of points

The outputs of the complex multipliers initiates sixteen stage 3 PPEA FFTs. This stage 3 FFT involves values that are 64 points apart. Hence for the next 64 clock instants when consecutive inputs arrive the inputs to the stage 3 FFT are generated and every consecutive set of 64 clock cycles provide consecutive inputs to the stage 3 PPEA FFT. When the 4032nd input arrives, the Stage 1 PPEA FFT involving that input is completed subsequently 4 stage 1 TWF multiplications are completed. The related 4 stage 2 PPEA FFT is completed and the respective stage 2 TWF multiplications are subsequently completed. These 16 outputs of the multipliers are needed to complete 16 stage 3 PPEA FFT. Hence when 4032nd input arrives this causes the completion of the first set of stage 3 PPEA FFT. The results from 16 stage 3 PPEA FFT are totally 64 in number which ideally needs 64 stage 3 TWF multiplications involving twiddle factors W_{256}^{r3c3} to be performed. Only 4 distinct twiddle factors are to be used of which one is unity. Since only 3 distinct twiddle factors are needed, 48 multipliers need to be used, but only 16 complex multipliers are used. A single Twiddle factor is used to perform the 16 multiplications. Initially multiplications are completed for the 2nd row elements and thereafter the third and fourth rows.

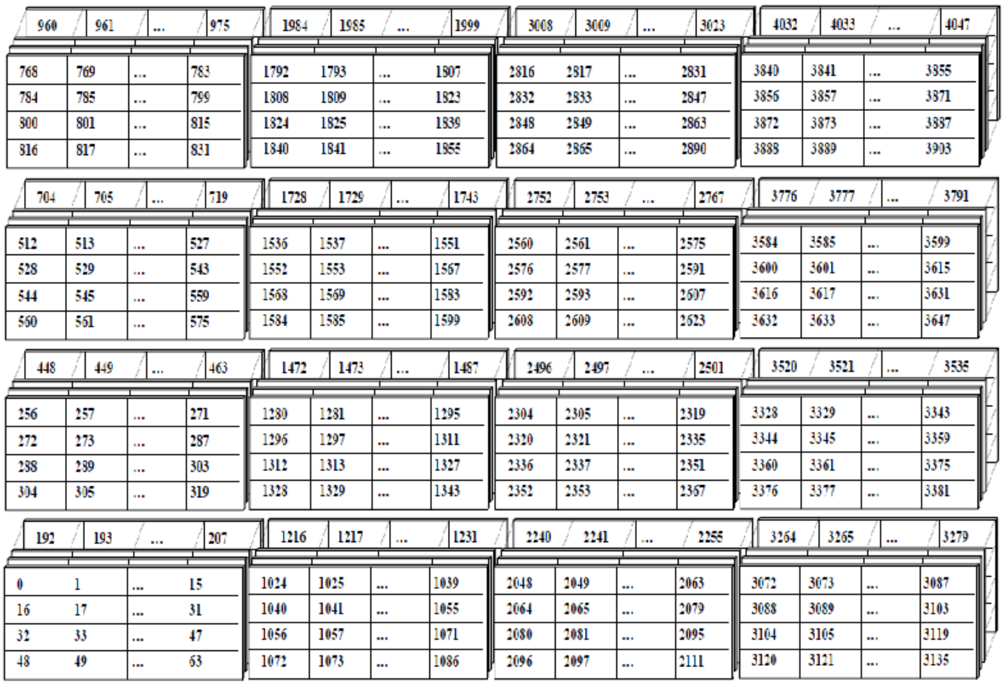


Figure 5: Stage 4 arrangement of PPEA FFT values, sixty four 4x16 matrix of points

In stage 4 the outputs of the complex multipliers from stage 3 initiates 16 stage 4 PPEA FFT. Stage 4 PPEA FFT is performed between points that are 16 points apart as shown in Fig. 5. In stage 3 64 twiddle factor multiplications should occur at the arrival of every input, but since only 16 multipliers are used only 32 values are available for stage 4 PPEA FFT. When the 4080th input arrives, the values for completion of the first Stage 4 PPEA FFT are available. At the arrival of 4080th input, the corresponding stage 1 PPEA FFT is completed, the respective 4 stage 1 TWF multiplications are completed and 4 stage 2 PPEA FFT and 16 stage 2 TWF multiplications are completed. Thereafter 16 stage 3 PPEA FFT, 16 stage 3 TWF multiplications are completed and finally 32 stage 4 PPEA FFTs are completed. This is followed by stage 4 TWF multiplication involving twiddle factor $W_{64}^{r4c4}$where r4 = 0 to 3 and c4= 0 to 15. Some of the multipliers used in stage 3 is reallocated, 4 gets allocated to stage 4. Similar to what happened in the earlier stages, PPEA FFT is followed by the twiddle factor multiplication.

At the arrival of the 4092nd input, stage 5 PPEA FFT is initiated which is between symbols that are 4 points apart, which corresponds to the column entries in Fig. 6. The stage 4 TWF multiplication outputs are inputs and FFT is completed after all the inputs have arrived and thereafter 4 multipliers are allocated from stage 2. The stage 5 TWF multiplication is done using twiddle factors $W_{16}^{r5c5}$where r5= 0 to 3 and c5 =0 to 3. Then the outputs of the stage 5 TWF multipliers are used to obtain the stage 6 PPEA FFT which is between points that are a point apart corresponding to adjacent row entries in Fig. 6.

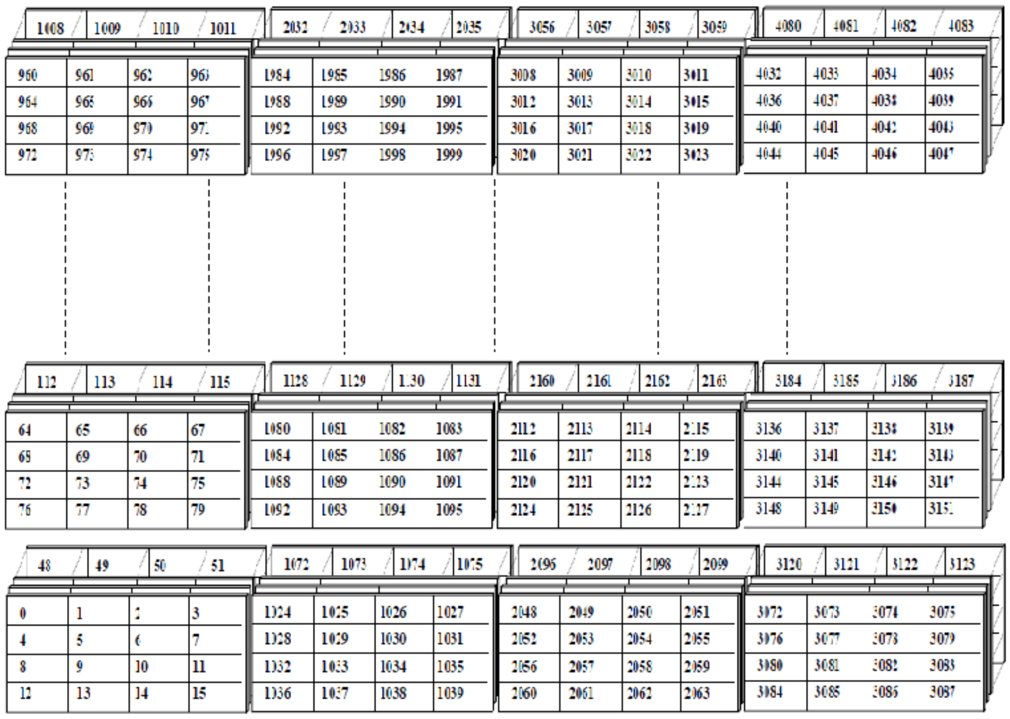


Figure 6: Stage 5 & 6 arrangement of PPEA FFT values, two hundred and fifty six 4 x 4 matrix of points

The inputs are virtually arranged in 4 x 1024 matrix as shown in Fig. 2 as the inputs arrive and 4 point PPEA FFT is performed. The stage 1, 4 point PPEA FFTs involve points [0, 1024, 2048, 3072], [1, 1025, 2049, 3073], up to [1023, 2047, 3071, 4095]. As stated earlier when the 3072nd input arrives the first FFT of the stage 1 is completed and the twiddle factor multiplications are performed. Each of the rows in the matrix shown in Fig. 2 is rearranged into 4 4x256 matrices as shown in Fig. 3. In every one of the 4x256 matrices shown in Fig. 3, 4 point PPEA FFT is started after the twiddle factor multiplications after the arrival of the 3072nd input. For examples the FFTs start out with the [0, 256, 512, 768], [1024, 1280, 1536, 1792], [2048, 2304, 2560, 2816] & [3072, 3328, 3584, 3840] and continue as a FFT gets completed in the stage 1 till it reaches [255, 511, 767, 1023], [1279, 1535, 1791, 2047], [2303, 2559, 2815, 3071] & [3327, 3583, 3843,4095].PPEA FFT is evaluated on arrival/availability and the hence the operation has to be repeated on arrival/availability of individual values in the matrix. As discussed earlier when the 3840th input arrives and stage-1 4 point PPEA FFT involving points [768, 1792, 2816, 3840] is completed. Thereafter the stage 1 twiddle factor multiplication and stage-2 4 point PPEA FFT involving points [0, 256, 512, 768], [1024, 1280, 1536, 1792], [2048, 2304, 2560, 2816] & [3072, 3328,3584, 3840] are completed. This makes 16 FFT outputs available for twiddle factor multiplication. The stage wise functions performed to obtain a 4096 point FFT is shown in Table 1.

After twiddle factor multiplications each of the 4 x 256 matrix are now arranged as 4 x 64 matrices as displayed in Fig. 4. As was explained and done in the earlier stages, the 4 point PPEA FFT is performed on every matrix arrangement as the previous stage FFT and twiddle factor multiplication is completed on a value in a specific location. When the 4032nd input arrives stage 1 FFT involving points [960, 1984, 3008, 4032] is completed and the subsequent twiddle factor multiplication is completed. The resulting 4 values cause 4 stage-2 FFTs involving points [192, 448, 704, 960], [1216, 1472, 1728, 1984], [2240, 2496, 2752, 3008] & [3264, 3520, 3776, 4032] to complete and the subsequent stage 2 twiddle factor multiplications are completed. These values in turn bring to completion the first stage 3 PPEA FFT. The subsequent stage 3 twiddle factor multiplications are undertaken. The resulting values are now arranged in 4x16 matrices. There are 64 4x16 matrices, as shown in Fig. 5, which are formed after the arrival of the 4080th input. The stage 1 PPEA FFT that is completed involves points in locations [1008, 2032, 3056, 4080]. The results multiplied by the corresponding twiddle factors and thereafter the stage 2 PPEA FFT involving [240, 496, 752, 1008], [1264, 1520, 1776, 2032], [2288, 2544, 2800, 3056] & [3312, 3568, 3824, 4080] is completed. After subsequent stage 2 twiddle factor multiplications, stage 3 PPEA FFT involving points [48, 112, 176, 240] till [3888, 3952, 4016, 4080] is completed. After the above mentioned 4 point PPEA FFT is completed and the resulting values propagate through the next stage the values cause the first set of PPEA FFTs to be completed in the stage 4. In the above matrices they constitute the first column of all the matrices. After the values are generated they are multiplied by the stage 5 twiddle factors. As the multiplications results are generated the values are stored in the 256 4x4 matrices as shown by Fig. 6.

Table 1: Stage wise functions – 4096 point FFT

FFT	**Factorization**	**Column-wise FFT**	**Twiddle factor**	**Row-wise FFT**
one 4096 point FFT	4 x 1024	1024, 4 point FFTs	W_{4096}^{rc}, r = 0 to 3, c = 0 to 1023	Four 1024 point FFTs
four 1024 point FFTs	4 x 256	256, 4 point FFTs for every 1024 point FFT	W_{1024}^{rc}, r = 0 to 3, c = 0 to 255	Four 256 point FFTs for every 1024 point FFT
sixteen 256 point FFT	4 x 64	64, 4 point FFTs for every 256 point FFT	W_{256}^{rc}, r = 0 to 3, c = 0 to 63	Four 64 point FFTs for every 256 point FFT
sixty four 64 point FFT	4 x 16	16, 4 point FFTs for every 64 point FFT	W_{64}^{rc}, r = 0 to 3, c = 0 to 15	Four 16 point FFTs for every 64 point FFT

two hundred and fifty six 16 point FFT	4 x 4	4, 4 point FFTs for every 16 point FFT	W_{16}^{rc}, r = 0 to 3, c = 0 to 3	Four 4 point FFTs for every 16 point FFT
One thousand and twenty four 4 point FFT	-	1, 4 point FFT	-	one 4 point FFT

These are stage 5 matrices which were formed after the arrival of the 4080th input and the subsequent PPEA FFT and twiddle factor multiplication in the subsequent stages. The values that arrive from stage 5 undergo PPEA FFT column-wise in the above mentioned matrices. For examples the points involved are [0,4,8,12], [16,20,24,28], [32,36,40,44] till [4080, 4084, 4088, 4092] initially and thereafter based on the availability of the successive values the FFT is extended to other columns of the matrices. When all the inputs have arrived, that is when the 4096th input arrives the last PPEA FFT of the stage 1 is completed and the generated values are processed through the successive PPEA FFT and multiplications. As was explained for the previous stages the arrival of the 4096th input triggers completion of PPEA FFTs in all the previous stages and the resulting values initiate the completion of FFT in stage 5. The use of PPEA FFTs and multipliers if unrestricted will allow 4 x 1024 values to be generated. But the use of twiddle factor multipliers are restricted to the values that contribute to the 'early' outputs and other values get access to the multipliers later to keep the resource requirements optimal. Since the outputs stream serially, it is unnecessary to produce all the outputs simultaneously and hence this measure. Thereafter the stage 6 PPEA FFT is performed row wise on the matrices formed in stage 5. The stage 6 PPEA FFT is applied to values which contribute to the 'early' outputs as all the inputs have arrived and to reduce the latency. In the initial stages these operations are applied to the inputs sequentially while after the 3rd stage the operations are applied on the basis of whether the value in the specific locations will transform into 'early' outputs.

The inputs as they arrive are processed through the stage 1 PPEA FFT and the resulting values are stored in the locations. Every input sample was allocated a sequence number, which was displayed in the matrix arrangement. Every input was virtually allocated a distinct location. The calculation of the PPEA FFT and twiddle factor multiplications is performed on values that contribute to the early outputsThe input sequence values after every subsequent processing that is, PPEA FFT and twiddle factor multiplication, are stored in distinct locations and hence it is easy to identify the outputs which are 'early'.

Twiddle factor Multiplications

The 4 point PPEA FFT is used as the inputs stream in one after the other and hence the part by part evaluation of contribution of every input to the FFT output is accumulated. This is followed by the twiddle factor multiplication by a factor W_{4096}^{r1c1}, where r1 = 0 to 3, c1 = 0 to 1023 and subsequently by row-wise 1024 point FFT. But as discussed above each 1024 point FFT has been calculated using 4 x 256 factorization. This approach was continued till the point where only 4 point FFT was needed. This approach is similar to radix 4 FFT except that the approach is based on the part by part evaluation and hence the reordering of inputs or outputs was not needed. This approach to computing 4096 point FFT using the 4 point PPEA FFT was organized into stages.

Twiddle factors used in different stages:

Stage 1: W_{4096}^{r1c1}, r1 = 0,1,2, 3 for every ‘c1’ & c1 = 0, 1,2,3,…., 1023.

Three twiddle factors are needed (since TWF is unity for r=0), r1 taking 4 values with c1 remaining a constant at a given instant of time and incrementing by 1 with time.

Stage 2: W_{1024}^{r2c2}, r2= 0, 1, 2, 3 for every ‘c2’ & c2 = 0,1,2,3,…., 255

The same three twiddle factors are needed for 4 multipliers each, r2 taking the 4 values with c2 remaining a constant at a given instant of time and incrementing by 1 with time.

Stage 3: W_{256}^{r3c3}, r3 = 0, 1, 2, 3 for every ‘c3’ & c3 = 0,1,2,3,….., 63

The same twiddle factor is needed for 16 multipliers, r3 taking the 4 values with c3 remaining a constant at a given instant of time and incrementing by 1 with time. The number of multipliers varies as the inputs arrive. The number of multipliers increases as stage 1 and stage 2 finish the TWF multiplication.

Stage 4: W_{64}^{r4c4}, r4 = 0, 1, 2, 3 for every ‘c4’ & c4 = 0,1,2,3,…., 15

The same twiddle factor is needed for 4 multipliers, r4 taking the 4 values with c4 remaining a constant at a given instant of time. The number of multipliers varies as the inputs arrive. The number of multipliers increases as stage 1 and stage 2 finish the TWF multiplication and it increases further after completion of stage 3.

Stage 5: W_{16}^{r5c5}, r5 = 0, 1, 2, 3 for every ‘c5’ & c5 = 0,1,2,3

The same twiddle factor is needed for 4 multipliers, r5 taking the 4 values with c5 remaining a constant at a given instant of time. The number of multipliers increases with completion of preceding stages.

The total number of distinct twiddle factors is 2047 for the stage 1 twiddle factor multiplication. For the consecutive stages from stage 2 to stage 5, (766, 190, 46, 10) distinct twiddle factors are needed. Stages 2 to 5 use the same twiddle factor for several multiplications since there are several similar sized matrices. During stage 1 multiplication 3 twiddle factors (one being unity) are needed simultaneously while in stage 2 12 twiddle factors are needed simultaneously to the multipliers. Out of these 12 twiddle factors there are only 3 distinct factors, meaning that 3 twiddle factors are used in 4 multipliers simultaneously. In stage 4, 48 twiddle factors are needed but only 3 twiddle factors are distinct but only 1 is used and hence one twiddle factor which is used by all the multipliers is read from the look up table. Thereafter the successive stages use only 4 multipliers needing only one twiddle factor to be read from the look up table. Here the value that is multiplied by the twiddle factor is prioritized based on whether it contributes to the ‘early’ outputs.

Results and Conclusion

When the 4096th input arrives, the last stage 1 PPEA FFT is completed and the results are forwarded to the 4 stage 1 TWF multipliers within a clock cycle. Stage 2 TWF multiplications are completed within a clock cycle and results are forwarded to 4 stage 2 PPEA symbol combiners which completes FFT within a clock period. Hence every subsequent stage's Symbol combiner(s) takes on clock period to complete the FFT as the forwarded input is the last input needed to complete the PPEA FFT and every stage's TWF multipliers take one clock cycle to complete the multiplication. There are 6 stages including the input stage, where 5 stages consist of the TWF multiplier sub-stage. Hence from the time of arrival of last input, the first output gets generated in after 11 clock cycles.

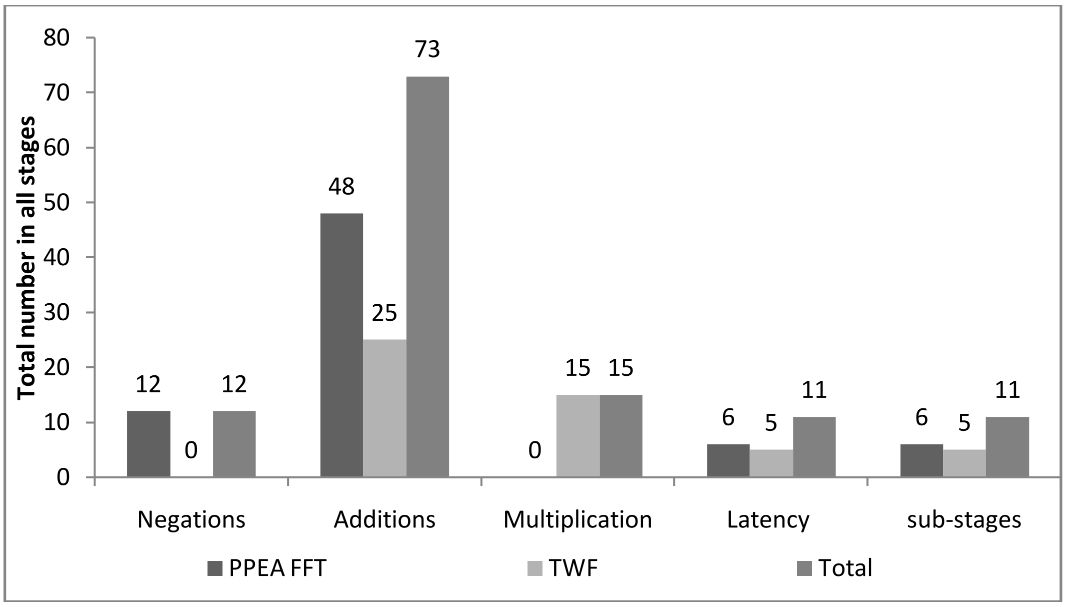


Figure 7: Arithmetic operations that affect Latency of PPEA FFT for N=4096

Latency Performance of Large Sized PPEA FFT

The reduction of latency of FFT/IFFT used in OFDM transceivers is the objective of this paper. Hence as the inputs stream in serially, the inputs are processed using 'divide and conquer' approach and the pipeline stages are established. Every pipeline stage except the last stage has PPEA Kernel(s) and TWF multipliers. When 4096 point PPEA FFT is implemented, it is implemented in stages as already discussed. Every sub-stage within a stage has finite number of arithmetic operations which affect latency. This means that after the arrival of the last input the arithmetic operations shown in Fig. 7 have to be completed before the first output is produced. The individual stages and sub-stages take only one clock period to process the last input is due to the usage of parallel sub-stages.

Twiddle factor multiplier blocks are sandwiched between PPEA FFT blocks and multipliers perform multiplications and pass the outputs as inputs to the successive PPEA FFT stage. The resources of the FPGA, especially the multipliers were distributed over all the stages, considering the ‘early’ outputs that have to be generated. Latency is dependent on the number of arithmetic operations needed to be performed to generate the first output after the arrival of the last input. The comparison of the latencies of the Radix 4 Single-path Delay commutator , Radix 2^2 Single-path Delay Feedback as discussed by Bin Zhou et al [17] and PPEA 8 point and PPEA 4 point is shown in Table 2 and the PPEA approach used in this paper showed an improved performance.

Table 2: Comparison of Latencies of Different Architectures In Clock Cycles

Architecture	**Point Size**	**Input data width**	**Latency (cycles)**
R4SDC	256	16	269
	1024	16	1041
R2^2SDF	256	16	270
	1024	16	1042
PPEA 4 point	4096	16	5007

The Pipelined radix – 2k feed-forward architecture as discussed by Mario Garrido Galvez et al (2013) which uses 16 parallel paths has offered the least latency and PPEA 8 point and PPEA 4 point offer lesser latencies than the former. The latency performance of PPEA FFT for N=512 is better than the latency performance of other architectures by a large margin. For example for R4SDC architecture 256 point FFT’s latency is 13 clock cycles after the arrival of the last input but the architecture implemented in this thesis has latency of just 5 clock cycles. Similarly the latency of PPEA 4 point for N=4096 is 11 clock cycles this much lesser than the R2^2SDF for N=1024. This improvement in latency performance comes at a cost of having higher resource utilization. When the PPEA 8 point FFT for N = 512 and PPEA 4 point FFT for N=4096 are compared with the contemporary architectures the latency offered in terms of clock cycles is considerably lesser as is explicit in the Table 2 and Table 3.

Table 3: Comparison of latencies of different architectures in seconds

Architecture	**Point Size**	**Input data width**	**Latency (µs)**
Pipelined radix -2k feed-forward (16 parallel)	1024	16	0.406
	4096	16	1.516
PPEA 8 point	512	16	0.0512
PPEA 4 point	4096	16	0.123

References

[1] Adiono, T. ; Mareta, R., “ Low latency parallel-pipelined Configurable FFT-IFFT 128/256/512/1024/2048 for LTE”, 4th International Conference

on Intelligent and Advanced Systems (ICIAS) publication, pp. 768-773, 2012

[2] Ahmed Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy, "Efficient FPGA implementation of FFT/IFFT Processor", International Journal Of Circuits, Systems And Signal Processing, Issue 3, Volume 3pp104-110, 2009

[3] Alexander A Petrovsky, Sergei L Shkredov, "Multi-Pipeline Implementations of Real-Time Vector DFT", Proceedings of the EUROMICRO systems on Digital System Design (DSD'04), IEEE Computer Society, 2004

[4] Babionitakis, K. ; Manolopoulos, K. ; Nakos, K. ; Reisis, D. ; Vlassopoulos, N. ; Chouliaras, V.A., "A High Performance VLSI FFT Architecture", 13th IEEE International Conference on Electronics, Circuits and Systems Proceedings, pp. 810-813, 2006

[5] Jesus Garcia, Juan A Michell, Gustavo Ruiz and Angel M Buron FPGA realization of a Split Radix FFT processor, Proceedings of SPIE, Vol. 6590 65900 pp.01-11,2007

[6] Bin Zhou, Yingning Peng, and David Hwang, "Pipeline FFT architectures Optimized for FPGA", International Journal of Reconfigurable Computing, Volume 2009, Article ID 219140, pp. 1-9, 2009

[7] Boopal. P. P, Garrido M., Gustafsson O., "A reconfigurable FFT architecture for variable length and multistreaming OFDM standards", IEEE conference Publications, IEEE International Symposium on Circuits and Systems (ISCAS, Page(s): 2066-2070,) 2013

[8] C. Yu, M.-H. Yen, P.-A. Hsiung, and S.-J. Chen, "A low-power 64-point pipeline FFT/IFFT processor for OFDM applications," IEEE Trans. on Consumer Electronics, vol. 57, no. 1, pp. 40-45, Feb. 2011

[9] Chao Cheng, Keshab K Parhi, "High-Throughput VLSI Architecture for FFT Computation", IEEE Transactions on Circuits and Systems-II:Express Briefs, Vol. 54, No.10, pp. 863-867, October 2007

[10] Chih-Peng Fan, Mau-Shih Lee, Guo-An Su, "A low multiplier and multiplication costs 256-point FFT implementation with simplified radix-2^4 SDF architecture", IEEE conference publications, IEEE Asia Pacific Conference on Circuits and Systems, (APCCAS 2006), Page(s): 1935-1938, 2006

[11] Chin-Long Wey ; Shin-Yo Lin ; Wei-Chien Tang ; Muh-Tien Shiue., "High-speed, Low Cost Parallel Memory-Based FFT processors for OFDM Applications", 14th IEEE International Conference on Electronics, Circuits and Systems (ICECS) proceedings pp. 783-787, 2007

[12] Chin-Long Wey ; Wei-Chien Tang ; Shin-Yo Lin, "Efficient Memory-Based FFT Architectures for Digital Video Broadcasting (DVB-T/H)" International Symposium on VLSI Design, Automation and Test (VLSI-DAT) proceedings, pp. 1-4, 2007

[13] DiptiSankar Das, Abhishek Mankar, N Prasad, K. K. Mahapatra, Ayas Kanta Swain Efficient VLSI Architectures of Split-Radix FFT using New

Distributed Arithmetic Ansuman International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1, pp 264 – 271 , March 2013

[14] Koushik Maharatna, Eckhard Grass, and Ulrich Jagdhold, "A 64-Point Fourier Transform Chip for High-speed Wireless LAN application using OFDM" IEEE Journal of Solid-State Circuits, Vol. 39, No. 3, pp. 484-493, March 2004

[15] Seungbeom Lee ; Sin-Chong Park., " Modified SDF Architecture for Mixed DIF/DIT FFT", Proceedings of International Conference on Communication Technology, pp.1-5, 2006

[16] Xu Peng ; Chen Jin Shu, "FPGA implementation of High Speed FFT algorithm" International Symposium on Intelligent Information Technology Application Workshops (IITAW'08) publication, pp. 781-784, 2008

[17] Mario Garrido Gálvez, J Grajal, M A. Sanchez and Oscar Gustafsson, Pipelined Radix-2(k) Feedforward FFT Architectures, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, (21), 1, 23-32, 2013