Fast Performance Pipeline Re-Configurable FFT Processor Based on Radix-2^2 for Variable Length *N*

Manish Bansal and Sangeeta Nakhate
MANIT EC Department, Bhopal, India
Email: {manishbansal2008; sanmanit}@gmail.com

Abstract—**This paper proposes fast performance reconfigurable pipeline variable points FFT processor design. This design is proposed for variable *N* points whose values can be 8, 16, 32, 64, 128, 256 and 512 sample points. Hence, the proposed reconfigurable design can be used for different points OFDM applications rather using different designs. The proposed implementation design uses Radix-2^2 Common Factor Algorithm (CFA) algorithm and SDF architecture. Radix-2^2 CFA algorithm reduces the number of twiddle factors compared to Radix-4 and Radix-2 and utilizes Radix-2 butterfly structure whose complexity is very low. SDF architecture uses less memory and utilizes multipliers fully compared to others. The frequency of proposed design is varied with the FFT points *N*. The proposed design is synthesized successfully using XST of Xilinx ISE 14.1 and simulated using ModelSim & MATLAB tool to verify results. Synthesized results of the proposed design for 512 points are 155.9 MHz max frequency, slice used 1795, slices Flip Flops used 1028 and LUTs used 3335 which are approximately 37% higher, 35%, 19%, 27% lesser in ratio than convention FFTs.**

Index Terms—**FFT, Radix-2^2 CFA, complex multiplier, SDF, OFDM**

I. Introduction

The Fast Fourier Transform (FFT) has been identified as a widely used algorithm in orthogonal frequency division multiplexing systems. Therefore, Fast Fourier transform is a fundamental building block used in DSP systems, with applications ranging from OFDM based Digital Modems, to Ultrasound, CT Image and RADAR algorithms as shown in Fig. 1. OFDM is a very flexible and efficient modulation technique to transmit data reliably on high transmission in the wireless and wire systems such as ADSL, IEEE802.11a, WPAN, WLAN and DVB-T [1]. FFT Implementation technique is fast and efficient than other techniques.

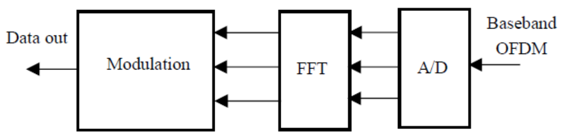


Fig. 1. Receiver OFDM transmission block diagram.

Manuscript received April 7, 2018; revised October 25, 2018; accepted October 25, 2018.

Corresponding author: Manish Bansal (manishbansal2008@gmail.com).

A reconfigurable computational scheme has attracted many researchers [2]. Hardware acts as a general hardware solution for implementing a variety of different computation within or across application domains which require performing FFT of length *N*. Implementing each FFT on dedicated IP is an overhead in silicon area of the chip. The novel approach is used to design variable length FFT processor to utilize the OFDM transmission. Some of the approaches to design FFT are memory-based, pipeline-based and general-purpose DSP. Memory based is most area efficient but it needs many computation cycles. Pipeline based architecture possesses regularity, modularity, local connection and high throughput rate with a lower clock frequency.

All implementations of FFT can be designed with different pipelined architectures - SDF (single path delay feedback), MDC (multipath delay commutator), and SDC (single path delay commutator) architectures. The single path delay feedback structure is more suitable than multipath delay and single path delay commutator architecture as: -

1) The SDF architecture is very convenient to implement the different length FFT.

2) The number of the required registers in SDF architecture is lesser than that in MDC and SDC structures [3].

The controller of proposed FFT design is easier than the other structures as overall architecture and all components are controlled by count clock signal only.

This proposed research presents the implementation of reconfigurable FFT for variable length *N* which can be used for 8, 16, 32, 64, 128, 256 and 512 points instead using the different processor for different point's applications. The radix impacts FFT processor. A small radix increases the count of twiddle factors and states simple structure of butterfly. A higher radix reduces the count of twiddle factors and states complex structure of butterfly. In the proposed processor, Radix-2^2 uses lesser twiddle factors compared to Radix-4, and butterfly structure based on Radix-2 that is the simplest structure.

This paper is structured as follows. Section II describes the proposed reconfigurable algorithm for variable length FFT. Section III describes the architecture of proposed reconfigurable Radix-2^2 FFT processor for different points. Section IV describes the comparison and the conclusion is given in the last section.

II. Proposed Reconfigurable Algorithm for Variable Length FFT

The Radix-2^2 Common Factor Algorithm (CFA) and butterfly structure are used to implement the variable length FFT processor. Thus, the proposed processor is designed in such a way that it works for 8, 16, 32, 64, 128, 512 points FFT calculation. The Proposed processor is implemented in pipeline single path delay feedback structure that is more convenient structure and reduces the FFT processor's complexity. Each FFT stage consists of butterfly structure. In this section, Radix-2^2 common factor algorithm, components, and the proposed FFT reconfigurable processor are described for variable length.

The definition of Discrete Fourier Transform (DFT) of size N is defined as [4]: -

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

where the twiddle factor W_N^{nk} is defined as $\exp(-j2\pi nk/N)$ and N represents the length of the sequence to be transformed. n is time domain ordinal. k is frequency domain ordinal. The Radix-2^2 algorithm is expressed using a common factor algorithm and 3-dimensional index mapping. The Radix-2^2 algorithm for N points is presented as [5].

Applying divide and conquer 3-D liner index mapping:

$$n=\frac{N}{2}n_1+\frac{N}{4}n_2+n_3 \tag{2}$$

$$k=k_1+2k_2+4k_3 \tag{3}$$

where N = 8, 16, 32, 64, 128, 256 and 512.

$$n_1,n_2=0,1 \text{ and } n_3=0,1,\cdots,\left(N/2^2\right)-1$$

$$k_1,k_2=0,1 \text{ and } k_3=0,1,\cdots,\left(N/2^2\right)-1$$

The common factor algorithm (CFA) form [4]:

$$X(k_1+2k_2+4k_3)=\sum_{n_3=0}^{(N/4)-1}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1}x\left(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right)W_N^{nk}$$

$$=\sum_{n_3=0}^{(N/4)-1}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1}x\left(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right)\times$$

$$W_N^{\left(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right)(k_1+2k_2+4k_3)} \tag{4}$$

The twiddle factor can be expressed as

$$W_N^{nk}=W_N^{\left(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right)(k_1+2k_2+4k_3)}$$
$$=\underbrace{(-1)^{n_1k_1}}_{\text{BU}}\underbrace{(-j)^{n_2k_1}}_{\text{PE}}\underbrace{(-1)^{n_1k_2}}_{\text{BU}}\underbrace{W_N^{n_3(k_1+2k_2)}W_{N/4}^{n_3k_3}}_{\text{PE}} \tag{5}$$

where n_1, n_2, and n_3 are the index terms of the input sample n and k_1, k_2, and k_3 are the index terms of the output sample k. $(-1)^{n_1k_1}$ and $(-1)^{n_1k_2}$ are butterfly elements. $(-j)^{n_2k_1}$ and $W_N^{n_3(k_1+2k_2)}$ are processing elements.

The detailed structure of the proposed pipeline Re-configurable FFT processor based on Radix-2^2 for variable length N is shown Fig. 2 that retains the Log$_2N$ number of stages. Shift register (SR) stores values. For N points, the 1st shift register stores $N/2$ values, the 2nd shift register stores $N/4$ values, the 3rd shift register stores $N/8$ values, so on and last shift register stores one value. Twiddle generator produces twiddle factors per $W_N^{n_3(k_1+2k_2)}$ based on n_3, k_1 and k_2 values. Swapper $(-j)^{n_2k_1}$ swaps signed real value and imaginary value then invert the sign of swapped imaginary value. Butterfly structure adds, subtracts and bypasses two input values. A complex multiplier multiplies twiddle factor value with input complex value

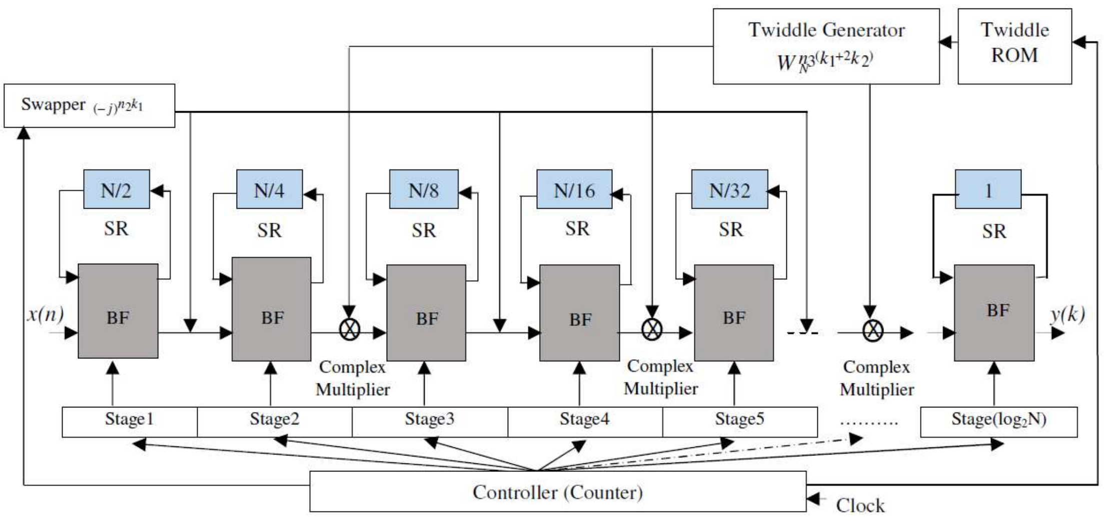


Fig. 2. Structure of the proposed reconfigurable FFT for N points.

Fig. 3 shows a signal flow graph (SFG) of the proposed reconfigurable Radix-2^2 FFT processor. It is to be noted that inputs are in normal order and outputs are in permuted (bit-reversed) order of inputs. First two stage calculations are based on n_1, n_2, n_3, k_1, k_2, k_3 which are calculated as per N-points. Similarly, next two stage calculations are based on n_1, n_2, n_3, k_1, k_2, k_3 which is calculated as per $N/4$-points and so on. First stage performs the operation between the nth and the (n+N/2)th points with butterfly structure, next stage performs the operation between the (n+N/2)th and the (n+N/4)th points and so on. This process runs continuously up to last stage $\log_2 N$ which performs the operation between the 1st and the 2nd point, the 3rd and the 4th point and so on.

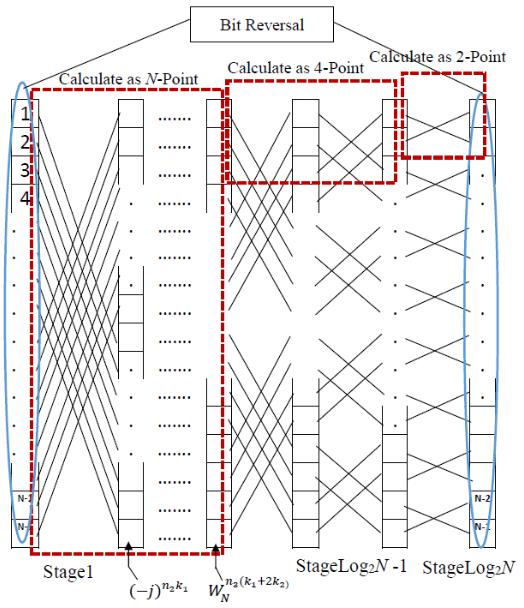


Fig. 3. SFG of the proposed reconfigurable FFT for N points.

Fig. 4 states the structure of butterfly (BU) that performs the calculation between the nth and the (n+ N/2)th value. The data inputted and flows from the input side to the shift registers when multiplexer (Mux) 1, 2, 3, 4 are at position '0' in the 1st N/2 cycles. The multiplexers turn to position '1' in the next N/2 cycles then butterfly begins to subtract/add operation to compute 2 points DFT with the stored shift registers data and next N/2 inputted data [6]. The subtracted outputs are stored in the same stage shift register, the 1st half numbers of added outputs are stored to the shift register of next stage and the 2nd half number of added outputs are subtracted/added with the 1st half number of added which is stored in the same stage. In this way, this flow persists from the present stage to next stage up to the final stage. Theses butterfly components are joined in pipelined structure to compute data in each clock.

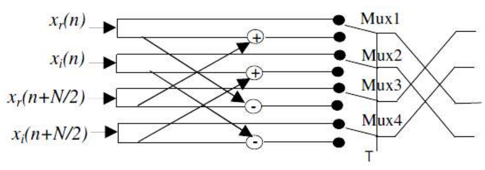


Fig. 4. Butterfly structure for N-Point

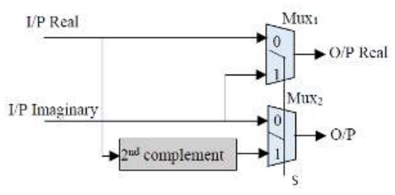


Fig. 5. Trivial (-j) multiplier (Swapper)

Fig. 5 Trivial (–j) multiplier performs swapping of real imaginary values and sign inversion of real value. The signal 'S' controls this swapping and the 2nd compliment invert real value sign. Whenever (–j) multiplication is required as per swapper $(-j)^{n_2k_1}$ value, multiplexers Mux_1 and Mux_2 switch to '1' then real value sign is inverted and imaginary signed value is swapped to real and inverted real signed value is swapped to imaginary. In this way, instead of doing (–j) multiplication, swapping & sign inversion is done. This approach enhances the frequency and reduces the logic and hardware.

III. Architectures of Proposed Reconfigurable Radix-2^2 FFT Processor for Different Points

The proposed FFT processor is designed in such a way that it automatically gets configured for variable points FFT. As shown in Fig. 2, all components - BFs, Stages, SRs, Twiddle generators, and Swappers are configured automatically based on the value of N and all operations are controlled by counter signal. Here the proposed reconfigurable FFT processor is designed based on the symmetry of different points FFTs that can be used for 8, 16, 32, 64, 128, 256, 512 points FFT instead of using different length FFT processors for different applications.

Table I shows components used in the proposed reconfigurable FFT design for variable points. As the value of N is provided, the number of stages, shift registers, twiddle generators, twiddle factors and swappers are automatically generated, configured and perform operations. In case of N points FFT, the proposed design generates $\log_2 N$ Number of stages and the 1st, 2nd, 3rd, 4th & other stages shift registers store N/2, N/4, N/8, N/16, …, 1 values respectively, Floor[($\log_2 N$−1)/2] number of twiddle generators, the 1st twiddle generator after the 2nd stage generates [((N/4)×3)–3] twiddle factors, the 2nd twiddle generator after the 4th stage generates [((N/4×4)×3)−3]×4 twiddle factors, the 3rd twiddle generator after the 6th stage generates [((N/4×4)×3)−3]×4×4 twiddle factors and similarly after others stages, Floor[($\log_2 N$)/2] number of swappers.

As shown in Table I, 16 points FFT have 4 stages, 2 swappers and 1 twiddle generator which generates 9 twiddle factors. Similarly, 32 points FFT have 5 stages, 2 swappers, and 2 twiddle generators. the 1st and the 2nd generator generates 21 and 12 twiddle factors respectively. These swappers and twiddle generators are PEs.

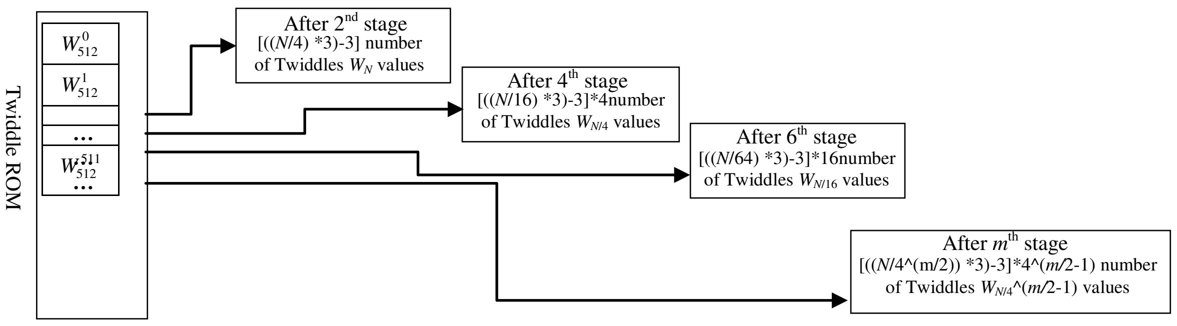
The PEs $(-j)^{n_2k_1}$ and $W_N^{n_3(k_1+2k_2)}$ after the 1st and the 2nd stages respectively are found out based on N for all the Nth positions, after the 3rd and the 4th stages respectively are found out based on N/4 for the (N/4)th positions and repeats the same PEs for every next 4 positions up to the last position, after the 5th and the 6th stages respectively are found out based on N/16 for the (N/16)th positions and repeats the same PEs for every next 16 positions up to the last position. The same way, PEs can be found at each position for next stages, later PEs at each position and each stage are shown in Fig. 8 and Fig. 9 for 16 points FFT and 32 Points FFT respectively will be discussed. Similarly, PEs at each position for others point's FFT can be found using parameters values in Table II.

TABLE I: COMPONENTS USED IN THE PROPOSED RECONFIGURABLE FFT BASED ON VARIABLE LENGTH N

N	Number of Stages	Number of Shift Register	Number of values Shift Registers store	Number of twiddle ROM	Number of twiddle generators	Number of twiddle factors	Number of Swappers
8	3	3	4+2+1	1	1	3	1
16	4	4	8+4+2+1	1	1	9	2
32	5	5	16+8+4+2+1	1	2	21+12	2
64	6	6	32+16+8+4+2+1	1	2	45+36	3
128	7	7	64+32+16+8+4+2+1	1	3	93+84+48	3
256	8	8	128+64+32+16+8+4+2+1	1	3	191+180+144	4
512	9	9	256+128+64+32+16+8+4+2+1	1	4	381+372+336+192	4

TABLE II: PROCESSING ELEMENTS PARAMETERS VALUES FOR THE PROPOSED RECONFIGURABLE FFT

PEs / N	After 1st Stage $(-j)^{n_2k_1}$	After 2nd Stage $W_N^{n_3(k_1+2k_2)}$	After 3rd Stage $(-j)^{n_2k_1}$	After 4th Stage $W_N^{n_3(k_1+2k_2)}$	After 5th Stage $(-j)^{n_2k_1}$	After 6th Stage $W_N^{n_3(k_1+2k_2)}$	After 7th Stage $(-j)^{n_2k_1}$	After 8th Stage $W_N^{n_3(k_1+2k_2)}$
8	$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1$ $N = 8$		No PE	No PE	No PE	No PE	No PE	No PE
16	$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 3$ $N = 16$		$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0$ $N = 4$	No PE	No PE	No PE	No PE	No PE
32	$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 7$ $N = 32$		$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1$ $N = 8$		No PE	No PE	No PE	No PE
64	$n_1, n_2 = 0, 1,$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 15$ $N = 64$		$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 3$ $N = 16$		$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0$ N=4	No PE	No PE	No PE
128	$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1,.., 31$ $N = 128$		$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 7$ $N = 32$		$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1$ N=8		No PE	No PE
256	$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 63$ $N = 256$		$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 15$ N=64		$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 3$ N=16		$n_1, n_2 = 0, 1, k_1, k_2 = 0, 1$ $n_3, k_3 = 0$ N=4	No PE
512	$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1,..,127$ $N = 512$		$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 31$ N=128		$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, .., 7$ N=32		$n_1, n_2 = 0, 1$ $k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1$ N=8	



Twiddle ROM
W_{512}^0
W_{512}^1
...
W_{512}^{511}
...
After 2nd stage
[((N/4) *3)-3] number of Twiddles W_N values
After 4th stage
[((N/16) *3)-3]*4number of Twiddles $W_{N/4}$ values
After 6th stage
[((N/64) *3)-3]*16number of Twiddles $W_{N/16}$ values
After mth stage
[((N/4^(m/2)) *3)-3]*4^(m/2-1) number of Twiddles $W_{N/4}$^(m/2-1) values

Fig. 6. Twiddle values fetched from twiddle ROM.

The twiddle ROM has 512 twiddle factors values from W_{512}^0 to W_{512}^{511} in the proposed processor. Even stages have the number of twiddle factors which are generated by twiddle generators and the value of these twiddle factors are fetched from twiddle ROM as shown in a Fig. 6 and Table III. These values are fetched from ROM based on symmetric $W_N^{k+N/2} = -W_N^k$ and periodicity $W_N^{k+N/2} = W_N^k$ properties. Some of the values are shown in Table III. This method is reducing area as even stages use the same values of twiddle ROM for different FFT points. The twiddle factor W_N^0 value wherever is required is bypassed rather than multiplying in the proposed processor as W_N^0 =1. These all techniques reduce hardware and increase the performance of the proposed design.

TABLE III: TECHNIQUE TO FIND TWIDDLE FACTORS FROM TWIDDLE ROM

PEs / N	After 2nd Stage $W_N^{n_3(k_1+2k_2)}$	After 4th Stage $W_N^{n_3(k_1+2k_2)}$	After 6th Stage $W_N^{n_3(k_1+2k_2)}$	After 8th Stage $W_N^{n_3(k_1+2k_2)}$
8	$W_{512}^{128}=W_8^2$ $W_{512}^{64}=W_8^1$ $W_{512}^{192}=W_8^3$	No PEs as there are only 3 stages in 8 points FFT		
16	$W_{512}^{64}=W_{16}^2$ $W_{512}^{32}=W_{16}^1$ $W_{512}^{96}=W_{16}^3$	No PEs as there are only 4 stages in 16 points FFT		
32	$W_{512}^{32}=W_{32}^2$ $W_{512}^{224}=W_{32}^{14}$ $W_{512}^{112}=W_{32}^7$	$W_{512}^{128}=W_8^2$ $W_{512}^{64}=W_8^1$ $W_{512}^{192}=W_8^3$	No PEs as there are only 5 stages in 32 points FFT	
64	$W_{512}^{240}=W_{64}^{30}$ $W_{512}^{120}=W_{64}^{15}$ $W_{512}^{360}=W_{64}^{45}$	$W_{512}^{64}=W_{16}^2$ $W_{512}^{32}=W_{16}^1$ $W_{512}^{96}=W_{16}^3$	No PEs as there are only 6 stages in 64 points FFT	
128	$W_{512}^{240}=W_{128}^{60}$ $W_{512}^{96}=W_{128}^{24}$ $W_{512}^{372}=W_{128}^{93}$	$W_{512}^{32}=W_{32}^2$ $W_{512}^{224}=W_{32}^{14}$ $W_{512}^{112}=W_{32}^7$	$W_{512}^{128}=W_8^2$ $W_{512}^{64}=W_8^1$ $W_{512}^{192}=W_8^3$	No PEs as there are only 7 stages in 128 points FFT
256	$W_{512}^{240}=W_{256}^{120}$ $W_{512}^{126}=W_{256}^{63}$ $W_{512}^{378}=W_{256}^{189}$	$W_{512}^{240}=W_{64}^{30}$ $W_{512}^{120}=W_{64}^{15}$ $W_{512}^{360}=W_{64}^{45}$	$W_{512}^{64}=W_{16}^2$ $W_{512}^{32}=W_{16}^1$ $W_{512}^{96}=W_{16}^3$	No PEs as there are only 8 stages in 256 points FFT
512	$W_{512}^{378}=W_{512}^{378}$ $W_{512}^{211}=W_{512}^{211}$ $W_{512}^{433}=W_{512}^{433}$	$W_{512}^{240}=W_{128}^{60}$ $W_{512}^{96}=W_{128}^{24}$ $W_{512}^{372}=W_{128}^{93}$	$W_{512}^{32}=W_{32}^2$ $W_{512}^{224}=W_{32}^{14}$ $W_{512}^{112}=W_{32}^7$	$W_{512}^{128}=W_8^2$ $W_{512}^{64}=W_8^1$ $W_{512}^{192}=W_8^3$

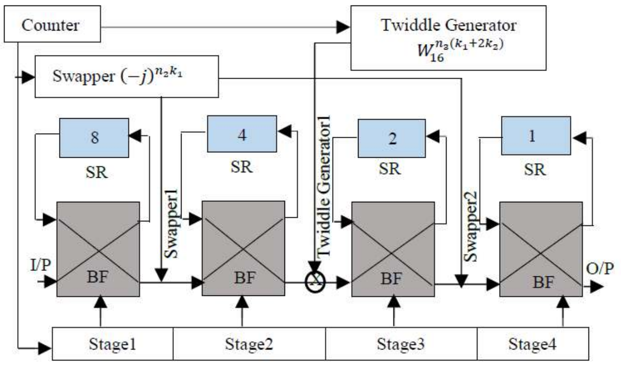


Fig. 7. Block diagram of the proposed reconfigurable FFT for 16 points

Fig. 7 shows a block diagram of 16 points FFT, the proposed design generates 4 shift registers, 4 stages, and 1 twiddle generator. The 1st stage registers stores 8 value, the 2nd stage register stores 4 value, the 3rd stage register stores 2 value and the 4th stage register stores 1 value as mentioned in Table I. The first stage, swapper $(-j)^{n_2k_1}$, the second stage and the twiddle factors $W_{16}^{n_3(k_1+2k_2)}$ operations are computed based on the 16 points. The third stage, the swapper $(-j)^{n_2k_1}$ and the fourth stage operations are computed based on 4 points samples as presented in Table II. Thus, the swapper $(-j)^{n2k1}$ after the 1st stage generates $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^1$, $(-j)^0$, $(-j)^0$, $(-j)^0$,$(-j)^1$,$(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^1$, $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^1$ and the twiddle generator $W_{16}^{n_3(k_1+2k_2)}$ after the 2nd stage generates twiddle factors W_{16}^0, W_{16}^0, W_{16}^0, W_{16}^0, W_{16}^0, W_{16}^2, W_{16}^4 W_{16}^6, W_{16}^0, W_{16}^1, W_{16}^2, W_{16}^3, W_{16}^0, W_{16}^3, W_{16}^6, W_{16}^9 and the swapper $(-j)^{n_2k_1}$ after the 3rd stage generates $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^0$,$(-j)^0$,$(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^0$, $(-j)^1$, $(-j)^1$, $(-j)^1$, $(-j)^1$.

Fig. 8 states the SFG of the 16 points proposed design reconfigurable processor. For the 16 points FFT results, the first 8 points are feed in the Stage 1 shift register by butterfly unit in every cycle. It takes 8 cycles and at the 9th cycle as $x(8)$ point is inserted into the Stage 1, butterfly unit starts to add and subtract between $x(0)$ & $x(8)$ points and this added output is feed in the Stage 2 shift register and subtracted output is feed in the Stage 1 shift register. At 10th cycle, $x(9)$ point entered into the Stage 1, the butterfly unit again begins to add and subtract between $x(1)$ & $x(9)$ points and the added output is feed in the Stage 2 register and subtracted is feed in the Stage 1 register. This flow continues up to the 12th cycle. At the 13th cycle as $x(12)$ point entered into the Stage 1, the butterfly again starts addition and subtraction between $x(4)$ & $x(12)$ point then this subtracted output is feet in the Stage 1 register and added value and the Stage 2 register values perform addition & subtraction then Stage 2 subtracted output is feedback in the Stage 2 register and the Stage 2 added output is feed in the Stage 3 register. This flow continues up to the 13th cycle. At the 15th cycle, as $x(14)$ point entered into the Stage 1, the Stage 3 butterfly unit also begins to add and subtract then this subtracted output is stored in the Stage 3 register and added output is stored in the Stage 4 register. At the 16th cycle, as $x(15)$ point entered into the Stage 1, the Stage 4 buttery unit also begins to add and subtract then this subtracted output is feed in the Stage 4 register and added output is the 1st output result y(0) of FFT output. This flow continuous up to the 31st cycle. Stage 4 additions outputs from 16th to 31st cycles are FFT results.

Swapping and twiddle multiplication operations are also performed at the clocks wherever it's required based on addresses as shown in Fig. 8 and Fig. 9. These operations are controlled by counter signal.

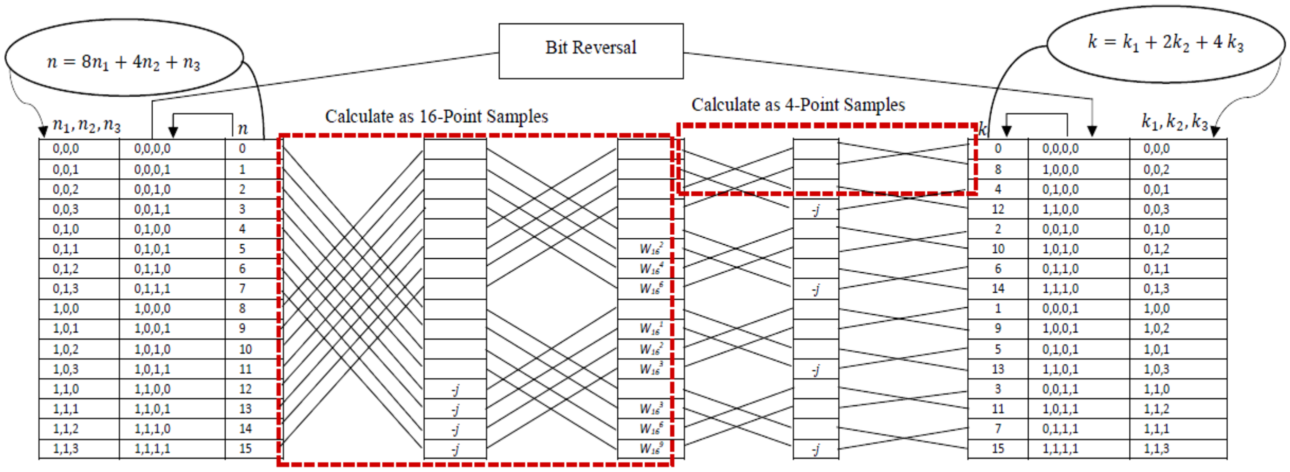


Fig. 8. SFG of proposed reconfigurable FFT for 16 points

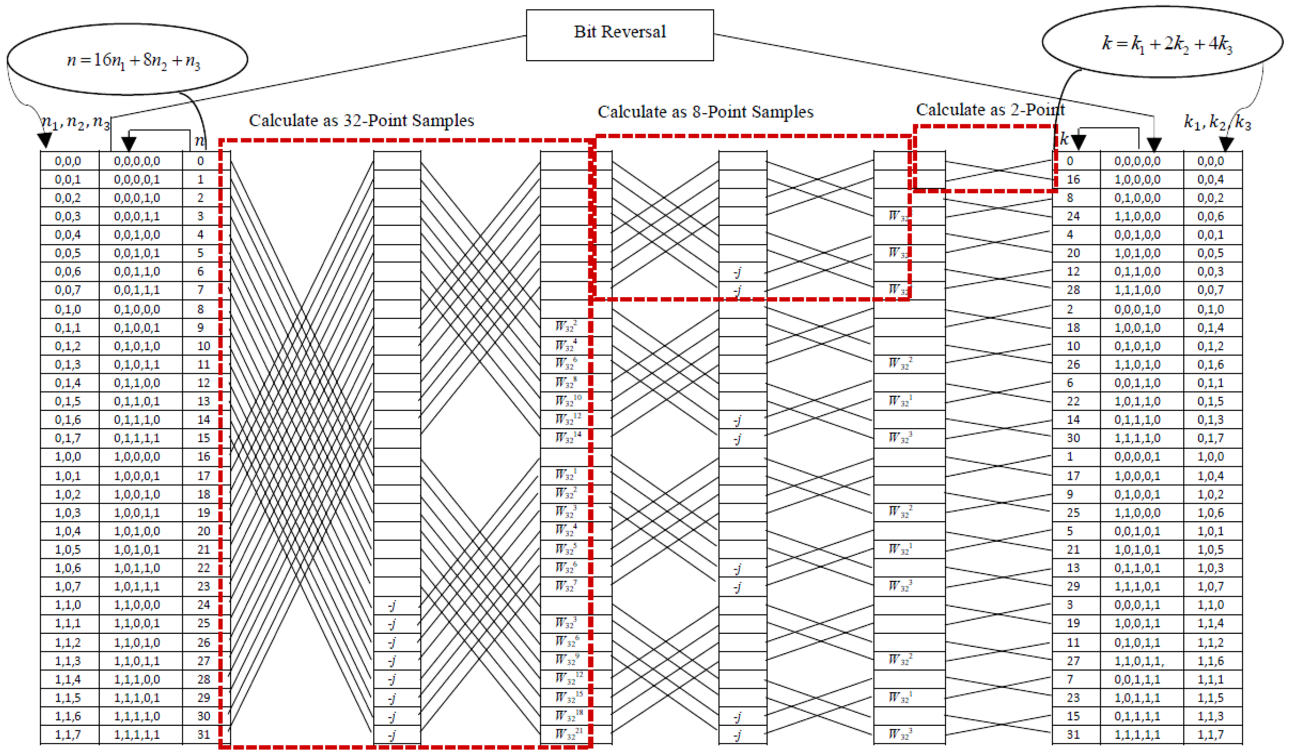


Fig. 9. SFG of proposed reconfigurable FFT for 32 points

IV. Comparison

The proposed pipeline re-configurable FFT processor based on Radix-2^2 for variable length *N* is designed in VHDL. It is simulated in MATLAB & ModelSim to validate the proposed design results and synthesized with xc7vx330t-3ffg1157 FPGA hardware.

Table IV shows the comparison of the proposed Reconfigurable Radix-2^2 and conventional FFT processors [8]-[11]. This table shows that the proposed design uses only one twiddle ROM which results in lesser twiddle factors, fewer multiplication operations, and less hardware. Table V shows implementation results of the proposed and conventional FFT processors [5, 4 and 7] between slices used, flip-flops used, LUT used and no. of GCLKs used. Fig. 10 shows the comparison between synthesized max frequencies of the proposed design for different points and conventional designs. The proposed design Max frequency for variable length is higher than conventional in ratio and max frequency of proposed design for variable length decreases as increasing number of points because of increased logic and complexity.

TABLE IV: COMPARISON OF DIFFERENT PROCESSORS

Processor	Number of Points	Method	Radix	Number of stages	Number of twiddle ROM	Memory
R2^2SDF [8]	256	Pipeline	Radix -2^2	8	8	256
R2^4SDF [9,10]	128	2 Parallel, Pipeline	Radix -2^4	7	7	128
R2^5SDF [11]	512	8 Parallel, Pipeline	Radix -2^5	9	9	512
Proposed reconfigurable Radix-2^2	16	Pipeline	Radix -2^2	4	1	16
	32			5		32
	64			6		64
	128			7		128
	256			8		256
	512			9		512

TABLE V: IMPLEMENTATION RESULTS

Parameters	FPGA [5]	FFT [4]	Vedic [7]	Proposed Reconfigurable Radix-2^2 FFT					
No. of Point (N)	1024	256	64	16	32	64	128	256	512
No. of slices used	3155	-	624	340	562	725	1042	1323	1795
No. of slices Flip Flops used	1514	-	-	284	410	538	697	851	1028
No. of 4 input LUTs used	5916	-	-	629	1031	1345	1937	2497	3335
Max Frequency (in MHz)	92.36	35.76	149.92	173.27	161.29	158.96	158.69	158.37	155.9
SQNR (in dB)	-	-	-	38.72	35.19	37.02	33.49	33.27	32.98

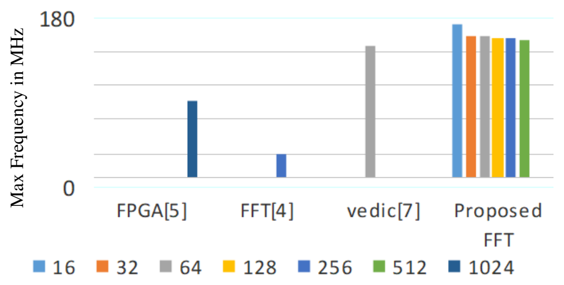


Fig. 10. Comparison of synthesis results of different architecture

V. CONCLUSIONS

The purpose of this research is to implement reconfigurable Radix-2^2 FFT processor that can be used for 8, 16, 32, 64, 128, 256, 512 points FFT instead of using different length FFT processor for different OFDM communication system FFT's application. The Radix-2^2 algorithm is used to reduce the number of twiddle factors, which causes the reduction in the area. The Radix-2 butterfly structure is used to reduce the complexity. This overall result increases speed as decrease number of multiplications, reusing twiddle factors value for others point FFT and passing input signal instead of multiplying the twiddle factor W_N^0 as W_N^0 value is 1. This approach increases the maximum frequency of the proposed reconfigurable FFT processor than conventional FFT processors.

REFERENCES

[1] A. Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy, "FPGA implementation of Radix-2^2 Pipelined FFT Processor," in *Proc. 3rd WSEAS Int. Symp. on Wavelets Theory and Applications in Applied Mathematics, Signal Processing & Modern Science*, 2009, pp. 109-114.

[2] Z. Wang, X. Liu, B. He, and F. Yu, "A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT," *IEEE Trans. VLSI Systems*, vol. 23, no. 5, pp. 973-977, 2015.

[3] J. Wang and L. A. Ronningen, "An implementation of pipelined radix-4 FFT architecture on FPGAs," *Journal of Clean Energy Technologies*, vol. 2, no. 1, pp. 101-103, Jan. 2014.

[4] C. P. Fan, M. S. Lee, and G. A. Su, "Efficient low multiplier cost 256-point FFT design with Radix-2^4 SDF architecture," *Journal of Engineering*, vol. 19, no. 2, pp. 61-74, 2008.

[5] K. Harikrishna, T. R. Rao, and V. A. Labay, "FPGA implementation of FFT algorithm for IEEE 802.16e (Mobile WiMAX)," *Int. Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 197-203, April 2011.

[6] W. H. Chang and T. Nguyen, "An OFDM-specified lossless FFT architecture," *IEEE Trans. on Circuits and Systems I*, vo. 53, no. 6, pp. 1235-1243, Jun. 2006.

[7] N. John and G. K. Sadanandan, "FPGA implementation of a novel efficient vedic FFT/IFFT processor for OFDM," *Int. Journal of Advanced Research in Electrical, Electronics and Instrumentation Engi.*, vol. 3, no. 3, pp. 7964-7972, Sep. 2014.

[8] U. Akshata and D. K. Gopika, "Hardware implementation of decimation in time FFT," *MIT Int. Journal of Electronics and Communication Engineering*, vol. 3, no. 1, pp. 39-42, Jan. 2013.

[9] J. Lee and H. Lee, "A high-speed parallel radix-2^4 FFT/IFFT processor for MB OFDM UWB system," *IEICE Trans. Fundamentals*, vol. E91–A, no. 4, April 2008.

[10] M. Shin and H. Lee, "A high-speed four-parallel radix-2^4 FFT/IFFT processor for UWB applications," in *Proc. IEEE Int. Symp. on Circuits and Systems*, May 2008, pp. 960-963.

[11] T. Cho and H. Lee, "A high-speed low-complexity modified FFT Radix - 2^5 processor for high rate WPAN applications," *IEEE Trans. on VLSI Systems*, vol. 21, no. 1, pp. 187-191, Jan. 2013.

[12] V. Sarada and T. Vigneswaran, "Reconfigurable FFT processor - A broader perspective survey," *Int. Journal of Engineering & Technology*, vol. 5, no. 2, pp. 949-956, May 2013.

[13] T. Cho, H. Lee, J. Park, and C. Park, "A high-speed low-complexity modified radix - 2^5 FFT processor for gigabit WPAN applications," in *Proc. IEEE Int. Symp. of Circuits and Systems*, May 2011, pp. 1259–1262.

[14] W. H. Chang and T. Nguyen, "Design and simulation of 64 point FFT using Radix - 4 algorithm for FPGA implementation," *Int. Journal of Engineering Trends and Technology*, vol. 4, no. 1-2, pp 109-113, Feb. 2013.

[15] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 21, no. 1, pp. 23-32, Jan. 2013.

[16] S. J. Huang and S. G. Chen, "A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs)," in *Proc. Int. Conf. on Green Circuits and Systems*, June 2010, pp. 9–13.

[17] W. Li, Y. Ma and L. Wanhammar, "Design and power measurement of different points FFT using Radix-2 algorithm for FPGA implementation," in *Proc. Int. Conf. of Electronics, Communication and Aerospace Technology*, April 2017, vol. 2, pp. 190-195.

[18] X. Cui and D. Yu, "Digital OFDM transmitter architecture and FPGA design," in *Proc. IEEE 8th Int. Conf. on ASIC*, Oct. 2009, pp. 477-480.

[19] M. Bansal and S. Nakhate, "High speed pipelined 64-point FFT processor based on Radix-2^2 for wireless LAN," in *Proc. IEEE 4th*

Int. Conf. on Signal Processing and Integrated Networks, Feb. 2017, pp. 607-612.

[20] S. He and M. Torkelson, "A new approach to pipeline processor," in *Proc. International Conference on Parallel Processing*, April 1996, pp. 766-770.

Manish Bansal received the B.E. degree in Electronics & Telecommunication and M.Tech. degree. His research interests include fast Fourier transform, signal processing, parallel computing, and VLSI.

Dr. Sangeeta Nakhate received the Ph.D. Her research interests include fast Fourier transform, signal processing, VLSI and microwave fields.