



Implementation of Split Radix Algorithm for 12-Point FFT & IFFT

P.Malyadri¹, D.Jayanayudu², G.Mahendra³

Associate Professor, Dept. of ECE, Prakasam Engineering College, Kandukur, Andhra Pradesh, India¹

Professor, Dept. of ECE, Sri Sathya Narayana Engineering College, Ongole, Andhra Pradesh, India²

PG Student [VLSI & ES], Dept. of ECE, Prakasam Engineering College, Kandukur, Andhra Pradesh, India³

ABSTRACT: Discrete Fourier transform (DFT) is widespread used in many fields of science and engineering. DFT is implemented with efficient algorithms categorized as Fast Fourier Transform. A fast algorithm is proposed for computing a length- $N=6^m$ DFT. The proposed algorithm is a blend of radix-3 and radix-6 FFT. It is $2^f \times 3^m$ variant of split radix and can be flexibly implemented a length DFT. Novel order permutation of sub-DFTs and reduction of the number of arithmetic operations enhance the practicability of the proposed algorithm. It inherently provides a wider choice of accessible FFT's lengths.

The proposed algorithm shows that its implementation requires less real operations as compared with the published algorithms. The pending update to system Verilog contains several new packages and functions. The new packages include support for both fixed-point and floating-point binary math. These fully Non-synthesizable packages will raise the level of abstraction in System Verilog. DSP applications, which previously needed an independent processor core, or required very difficult manual translation, can now be performed within your system Verilog source code. In addition, Schematic-based DSP algorithms can now be translated directly to System Verilog.

KEY WORDS: Discrete Fourier transform (DFT), Fast Fourier transform (FFT), Inverse Fast Fourier transform (IFFT), general split radix, radix 3/6, System Verilog language.

I.INTRODUCTION

Discrete Fourier Transform (DFT) is one of the most important tools used in almost all fields of science and engineering. DFT can be implemented with efficient algorithms generally classified as Fast Fourier transforms (FFT). The most widely used approaches are so-called the algorithms for 2^m , such as radix-2, radix 4 and split radix FFT (SRFFT). Considerable researches have carried out and resulted in the rapid development on this class of algorithms. Simultaneously, the researches on the algorithms for computing length- $N=k^m$ DFT have resulted in the presentation of the methods for and $k=3$ and $k=6$.

Due to the poor efficiency, the algorithms for k^m are of trivial practical meanings when $k \neq 2$. However, there exist many applications in which the sequence lengths are 3^m or 6^m . The idea of this letter is to develop a useful algorithm for length $N=6^m$ DFT. The available published algorithms are reported in; it seems that the general split radix algorithm is more adequate for the length- DFT. In this letter, we propose an algorithm based radix-6 approach. The algorithm is implemented with more efficient than the reported ones. Its computational Complexity is approximately equal to the equation given as $4.071N \log_2 N - 5.61N + 33.555 \log_2 N - 130.992$ which is close to that of standard SRFFT (The complexity of SRFFT is $4N \log_2 N - 6N + 8$). The proposed algorithm is a radix 3/6 algorithm and uses base $(1, j)$. The algorithm decomposes a DFT of size $N=6^m$ into one length- $N/3$ and four length- $N/6$ sub DFTs. The flexibility of the decomposition enables the algorithm is competent at the implementation of a non-power-of-six DFT, while its length can exactly divided by 6. Appropriate permutations are used for sub DFTs input sequences to reduce the computational intension.



International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

II. LITERATURE SURVEY

a. Radix 2/8 FFT algorithm for length $qx2^m$

A new radix-2/8 Fast Fourier Transform (FFT) algorithm have been proposed for computing the Discrete Fourier transform of an arbitrary length $N= qx2^m$, where m is an odd integer. It reduces substantially the operations such as data transfer, address generation, and twiddle factor evaluation or access to the lookup table, which contribute significantly to the execution time of FFT algorithms. It is shown that the arithmetic complexity (multiplications, additions) of the proposed algorithm is, in most cases, the same as that of the existing split-radix FFT algorithm. The basic idea behind the proposed algorithm is the use of a mixture of radix-2 and radix-8 index maps. The algorithm is expressed in a simple matrix form, thereby facilitating an easy implementation of the algorithm, and allowing for an extension to the multidimensional case. For structural complexity, the important properties of the Cooley–Tukey approach such as the use of the butterfly scheme and in-place computation are preserved by the proposed algorithm. It is suitable only for DFT of sequence length $N=qx2^m$.

b. Radix 2/16 FFT algorithm for length $qx2^m$

A radix-2/16 decimation-in-frequency (DIF) fast Fourier transforms (FFT) algorithm and its higher radix version, namely radix-4/16 DIF FFT algorithm, have been proposed by suitably mixing the radix-2, radix-4 and radix-16 index maps, and combing some of the twiddle factors [3]. It is shown that the proposed algorithms and the existing radix-2/4 and radix-2/8 FFT algorithms require exactly the same number of arithmetic operations (multiplications and additions). By using this technique, it can be shown that all the possible split-radix FFT algorithms of the type radix- $2r/2rs$ for computing a 2^m DFT require exactly the same number of arithmetic operations. This algorithm is suitable only for sequence of length $N=2^m$, m is integer.

III. IMPLEMENTATION OF PROPOSED RADIX 3/6 ALGORITHM

A new radix-6 FFT algorithm suitable for multiply-add instruction have been proposed. The new radix-6 FFT algorithm requires fewer floating-point instructions than the conventional radix-6 FFT algorithms on processors that have a multiply-add instruction. Techniques to obtain an algorithm for computing radix-6 FFT with fewer floating-point instructions than conventional radix-6 FFT algorithms have been proposed. The number of floating-point instructions for the new radix-6 FFT algorithm is compared with those of conventional radix-6 FFT algorithms on processors with multiply-add instruction.

The definition of the Discrete Fourier Transform is given by

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk} \quad (1)$$

Where

$$W_N = e^{-j2\pi/N}, j = \sqrt{-1}, \text{ the length of sequence}$$

$\{x_n\}$ is assumed as an integer which is divisibly by six. For lengths N of DFT, powers-of-six would be best for the proposed algorithm. Obviously, the DFT can be divided into three length- $N/3$ sub-DFTs. In order to derive a best possible algorithm, we continue to decompose the three sub-DFTs. Due to no scaling factor in front of it; the first sub-DFT should be let as-is and directly go into the recursive decomposition of the next stage. The other two sub-DFTs are divided into four sub-DFTs of length- $N/6$. Actually, if the length of a DFT can be divided by 6, the DFT can be definitely decomposed by the algorithm. The generalized length- N can be assumed as the $N= 2^r x 3^m$, where as $r \geq m-1$. The decomposition of a DFT of size $N= 2^r x 3^m$ denoted by

$$X_K = \sum_{n=0}^{N-1} x_{3n} W_N^{nk} + W_{2^r}^k W_{3^m}^k \sum_{n=0}^{N-1} x_{6n} + 2^r + 3^m W_{N/6}^{nk} + W_{3^m}^k \sum_{n=0}^{N-1} x_{6n} + 2^r W_{N/6}^{nk} + W_{3^m}^{-k} \sum_{n=0}^{N-1} x_{6n} - 2^r W_{N/6}^{nk} + W_{2^r}^{-k} W_{3^m}^{-k} \sum_{n=0}^{N-1} x_{6n} - 2^r - 3^m W_{N/6}^{nk} \quad (2)$$

Where the four length- $N/6$ sub DFTs are reordered. To simplify the description, (2) can be expressed

$$X_k = A_k + W_{2^r}^k W_{3^m}^k B_k + W_{3^m}^k C_k + W_{3^m}^{-k} E_k + W_{2^r}^{-k} W_{3^m}^{-k} F_k \quad (3)$$

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

Where

$$\begin{aligned}
 A_k &= \sum_{n=0}^{N/3-1} x_{3n} W_{\frac{N}{3}}^{nk} \\
 B_k &= \sum_{n=0}^{N/6-1} x_{6n} + 2^r + 3^m W_{\frac{N}{6}}^{nk} \\
 C_k &= \sum_{n=0}^{N/6-1} x_{6n} + 2^r W_{\frac{N}{6}}^{nk} \\
 E_k &= \sum_{n=0}^{N/6-1} x_{6n} - 2^r W_{\frac{N}{6}}^{nk} \\
 F_k &= \sum_{n=0}^{N/6-1} x_{6n} - 2^r - 3^m W_{\frac{N}{6}}^{nk}
 \end{aligned} \tag{4}$$

In (3), $W_{2^r}^k W_{3^m}^k B_k$ and $W_{2^r}^{-k} W_{3^m}^{-k} F_k$ can be treated in pairs, since $W_{2^r}^k W_{3^m}^k$ and $W_{2^r}^{-k} W_{3^m}^{-k}$ is a conjugate-pair. In the similar way, $W_{3^m}^k C_k$ and $W_{3^m}^{-k} E_k$ can be handled with in pairs. The direct implementation of (3) performs many unnecessary operations, since the computations of $X_k, X_{2N/6+k}, X_{4N/6+k}, X_{N/6+k}, X_{3N/6+k}$ and $X_{5N/6+k}$ turn out to share many calculations each other. In particular, if we add $N/6$ to K , the size- DFT are not changed (because they are periodic in), while the size- $N/3$ DFT is unchanged if we add $2N/6$ to K . So, the only things that changes are $W_{2^r}^k W_{3^m}^k, W_{2^r}^{-k} W_{3^m}^{-k}, W_{3^m}^k$ and $W_{3^m}^{-k}$ terms. In order to reduce the number of the operations, the following six identities are necessary,

$$X_k = A_k + (w_{2^r}^k w_{3^m}^k B_k + w_{2^r}^{-k} w_{3^m}^{-k} F_k) + (w_{3^m}^k C_k + w_{3^m}^{-k} E_k) \tag{5}$$

$$X_{2N/6+k} = A_k + (w_3^{2r} w_{2^r}^k w_{3^m}^k B_k + w_3^{-2r} w_{2^r}^{-k} w_{3^m}^{-k} F_k) + (w_3^{2r} w_{3^m}^k C_k + w_3^{-2r} w_{3^m}^{-k} E_k) \tag{6}$$

$$X_{4N/6+k} = A_k + (w_3^{2r+1} w_{2^r}^k w_{3^m}^k B_k + w_3^{-2r+1} w_{2^r}^{-k} w_{3^m}^{-k} F_k) + (w_3^{2r+1} w_{3^m}^k C_k + w_3^{-2r+1} w_{3^m}^{-k} E_k) \tag{7}$$

$$X_{N/6+k} = A_{N/6+k} w_3^{2r-1} w_{2^r}^k w_{3^m}^k B_k + w_3^{-2r-1} w_{2^r}^{-k} w_{3^m}^{-k} F_k) + (w_3^{2r-1} w_{3^m}^k C_k + w_3^{-2r-1} w_{3^m}^{-k} E_k) \tag{8}$$

$$X_{3N/6+k} = A_{N/6+k} (w_{2^r}^k w_{3^m}^k B_k + w_{2^r}^{-k} w_{3^m}^{-k} F_k) + (w_{3^m}^k C_k + w_{3^m}^{-k} E_k) \tag{9}$$

$$X_{5N/6+k} = A_{N/6+k} (w_3^{2r} w_{2^r}^k w_{3^m}^k B_k + w_3^{-2r} w_{2^r}^{-k} w_{3^m}^{-k} F_k) + (w_3^{2r} w_{3^m}^k C_k + w_3^{-2r} w_{3^m}^{-k} E_k) \tag{10}$$

A complete output $X \{k\}$ set can be obtained if we let range from 0 to $N/6 - 1$ in the above six equations.

We now summarize the scheme of the proposed radix-3/6 FFT algorithm. The initial input sequence $\{x_n\}$ of length- N is decomposed into five sub-sequences. This process is repeated successively for each of new sub-sequences, until the sizes of all sub DFTs are indivisible by 6. Figs. 1–3 illustrate the flow graph of 3, 6 and 12-point radix 3/6 algorithm (2-points and 4-points FFT can be performed with SRFFT).

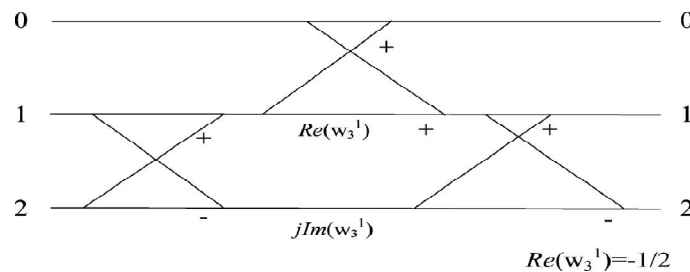


Fig.1: Flow graph for 3-point FFT

The 3-points DFT requires 4 real multiplications and 12 real additions (some algorithms assume that a 3-points DFT is calculated with 2 real multiplications and 12 real additions since one need not multiply $1/2$ and the multiplication by $1/2$ can be evaluated with bit shift).

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

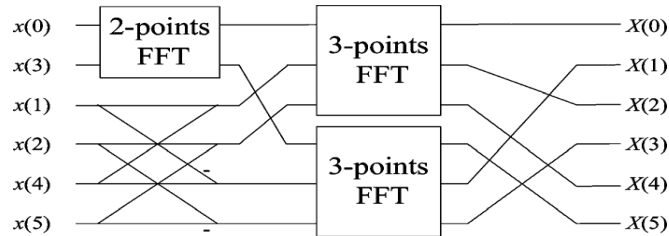


Fig.2: Flow Chart For 6-point FFT

The initial input sequence $\{x_n\}$ of length- N is decomposed into five sub-sequences. This process is repeated successively for each of new sub-sequences, until the sizes of all sub DFTs are indivisible by 6.

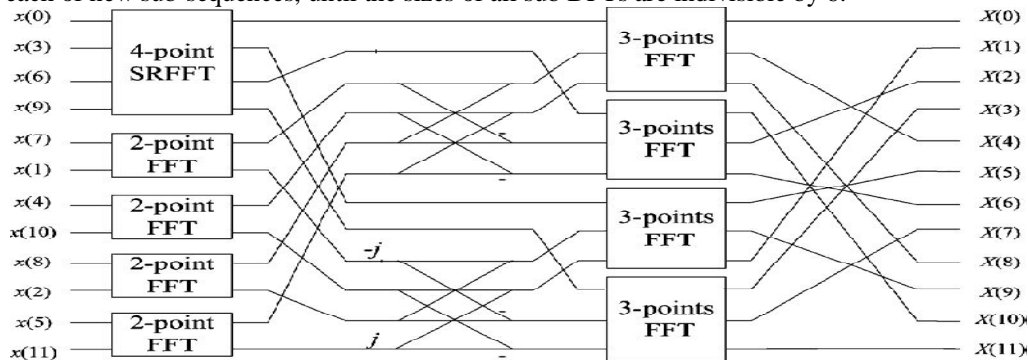


Fig. 3: Flow Chart For 12-Point radix- 3/6 FFT

The Radix-3/6 DIF FFT can be derived as follows

$$\begin{aligned}
 X(k) &= \sum_{n=0}^{N-1} x_n W_N^{nk} \\
 &= \sum_{n=0}^{\frac{N}{3}-1} x(3n)W_N^{3nk} + \sum_{n=0}^{\frac{N}{3}-1} x(3n+1)W_N^{(3n+1)k} + \sum_{n=0}^{\frac{N}{3}-1} x(3n+2)W_N^{(3n+2)k} \\
 &= \sum_{n=0}^{\frac{N}{3}-1} x(3n)W_N^{3nk} + W_N^k \sum_{n=0}^{\frac{N}{3}-1} x(3n+1)W_N^{3nk} + W_N^{2k} \sum_{n=0}^{\frac{N}{3}-1} x(3n+2)W_N^{3nk} \\
 &= \sum_{n=0}^{\frac{N}{3}-1} x(3n)W_{N/3}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{3}-1} x(3n+1)W_{N/3}^{nk} + W_N^{2k} \sum_{n=0}^{\frac{N}{3}-1} x(3n+2)W_{N/3}^{nk} \\
 &= P(k) + W_N^k Q(k) + W_N^{2k} R(k)
 \end{aligned}$$

Each of the sums, P (k), Q (k), and R (k), in is recognized as an N/3-point DFT. The transform X (k) can be broken into three parts as shown in equation (11).

$$\begin{aligned}
 X(k) &= P(k) + W_N^k Q(k) + W_N^{2k} R(k) \\
 X\left(k + \frac{N}{3}\right) &= P(k) + W_N^{k+\frac{N}{3}} Q(k) + W_N^{2\left(k+\frac{N}{3}\right)} R(k) \\
 &= P(k) + W_N^{\frac{1}{3}} W_N^k Q(k) + W_N^{\frac{2}{3}} W_N^{2k} R(k) \\
 X\left(k + \frac{2N}{3}\right) &= P(k) + W_N^{k+\frac{2N}{3}} Q(k) + W_N^{2\left(k+\frac{2N}{3}\right)} R(k) \\
 &= P(k) + W_N^{\frac{2}{3}} W_N^k Q(k) + W_N^{\frac{1}{3}} W_N^{2k} R(k)
 \end{aligned} \tag{11}$$

$$k = 0, 1, 2, \dots, \frac{N}{3} - 1$$

$$W_3^1 = \exp\left(-\frac{j2\pi}{3}\right) = -\frac{1}{2} - \frac{\sqrt{3}}{2}j \tag{12}$$

$$W_3^2 = \exp\left(-\frac{j2\pi \times 2}{3}\right) = -\frac{1}{2} + \frac{\sqrt{3}}{2}j \tag{13}$$



International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

The decomposition in the proposed algorithm is conducted recursively until the lengths of all sub DFTs cannot be exactly divided by 6. In general, there are only 1 the first special butterfly (if $r \geq 1$ and $m \geq 1$), 1 the second special case butterfly (if $r \geq 2$ and $m \geq 1$), 1 the third special case butterfly and 1 the fourth special case butterfly (if $r \geq 3$ and $m \geq 1$). The total number of the fifth and sixth type of butterflies is $2^{r-1}-4$. Thus, the arithmetic complexity of the proposed algorithm can be given in below equation (14).

$$M_N = \begin{cases} \frac{M_{N/3} + 4M_{N/6} + 8N}{3-8} & r = 1, m \geq 1 \\ \frac{M_{N/3} + 4M_{N/6} + 8N}{3-16} & r = 2, m \geq 1 \\ \frac{M_{N/3} + 4M_{N/6} + 8N}{3-24} & r \geq 3, m \geq 1 \end{cases}$$

$$A_N = \begin{cases} \frac{A_{N/3} + 4A_{N/6} + 20N}{3-8} & r = 1, m \geq 1 \\ \frac{A_{N/3} + 4A_{N/6} + 20N}{3-16} & r = 2, m \geq 1 \\ \frac{A_{N/3} + 4A_{N/6} + 20N}{3-2^{r+1}-8} & r \geq 3, m \geq 1 \end{cases} \quad (14)$$

IV. RESULTS AND DISCUSSION

The 12 point DFT sequence has been implemented in System Verilog and simulated using Modelsim Version 6.4.

Message	Value
/Twelve_Point_main/in1r	1
/Twelve_Point_main/in1i	0
/Twelve_Point_main/in2r	2
/Twelve_Point_main/in2i	0
/Twelve_Point_main/in3r	3
/Twelve_Point_main/in3i	0
/Twelve_Point_main/in4r	4
/Twelve_Point_main/in4i	0
/Twelve_Point_main/in5r	5
/Twelve_Point_main/in5i	0
/Twelve_Point_main/in6r	6
/Twelve_Point_main/in6i	0
/Twelve_Point_main/in7r	7
/Twelve_Point_main/in7i	0
/Twelve_Point_main/in8r	8
/Twelve_Point_main/in8i	0
/Twelve_Point_main/in9r	9
/Twelve_Point_main/in9i	0
/Twelve_Point_main/in10r	10
/Twelve_Point_main/in10i	0
/Twelve_Point_main/in11r	11
/Twelve_Point_main/in11i	0
/Twelve_Point_main/in12r	12
/Twelve_Point_main/in12i	0

Fig.4: Simulation result of Radix-3/6 12-point DFT input Sequence

The, Fig.4 shows the input sequence of radix-3/6 algorithm for 12-point FFT i.e., $\{x_n\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ applied to the Modelsim for simulation.

Message	Value
/Twelve_Point_main/out1r	78
/Twelve_Point_main/out1i	0
/Twelve_Point_main/out2r	-6
/Twelve_Point_main/out2i	22.392
/Twelve_Point_main/out3r	-6
/Twelve_Point_main/out3i	10.392
/Twelve_Point_main/out4r	-6
/Twelve_Point_main/out4i	6
/Twelve_Point_main/out5r	-6
/Twelve_Point_main/out5i	3.464
/Twelve_Point_main/out6r	-6
/Twelve_Point_main/out6i	1.608
/Twelve_Point_main/out7r	-6
/Twelve_Point_main/out7i	0
/Twelve_Point_main/out8r	-6
/Twelve_Point_main/out8i	-1.608
/Twelve_Point_main/out9r	-6
/Twelve_Point_main/out9i	-3.464
/Twelve_Point_main/out10r	-6
/Twelve_Point_main/out10i	-6
/Twelve_Point_main/out11r	-6
/Twelve_Point_main/out11i	-10.392
/Twelve_Point_main/out12r	-6
/Twelve_Point_main/out12i	-22.392

Fig.5: Simulation output for 12-point input Sequence

The, Fig.5 shows the output sequence $X_k = \{78, -6 + 22.3923i, -6 + 10.39i, -6 + 6i, -6 + 3.464i, -6 + 1.6i, -6, -6 - 1.6i, -6 - 3.4i, -6 - 6i, -6 - 10.39i, -6 - 22.3923i\}$ for the given input sequence.

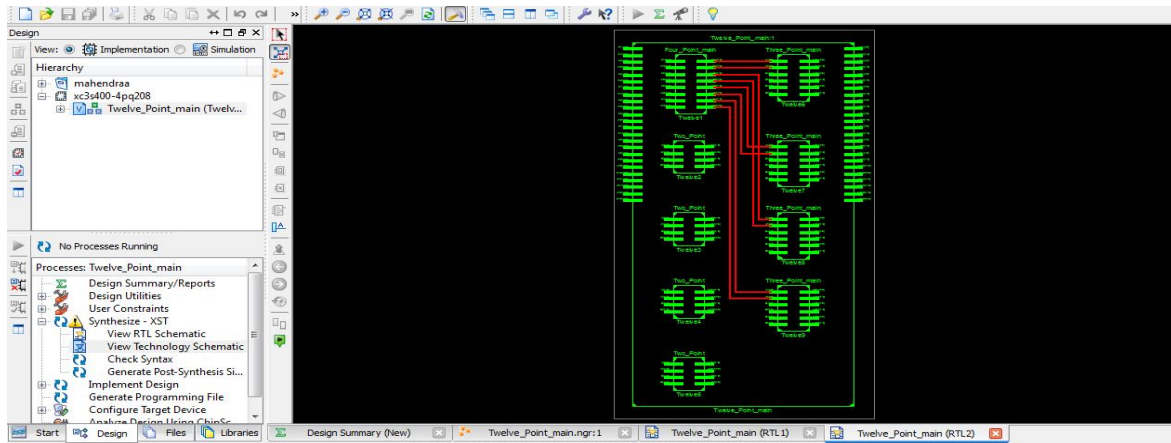
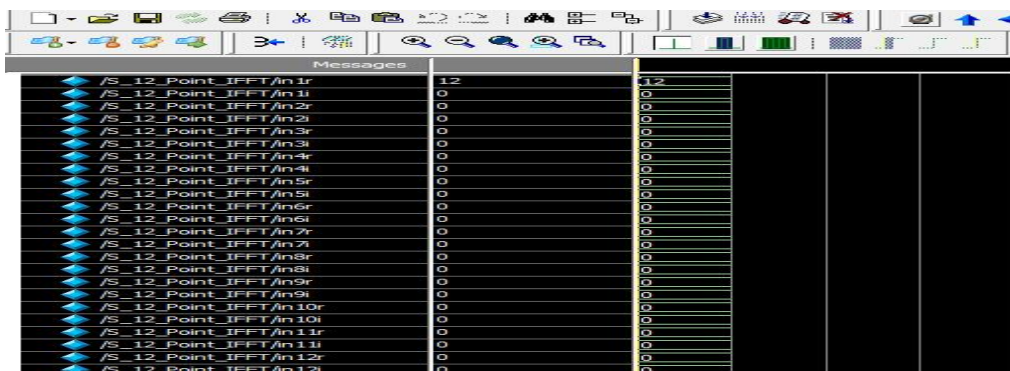


Fig.6: RTL Schematic for 12-point FFT radix 3/6 algorithm

The above figure.6 is the RTL (Register Transfer Level) Schematic for 12 point input sequence using radix 3/6 algorithm generated in Xilinx 9.1i. The schematic contains one 4-point SRFFT, four 2-point FFT and three 3-point FFTs butterfly blocks.

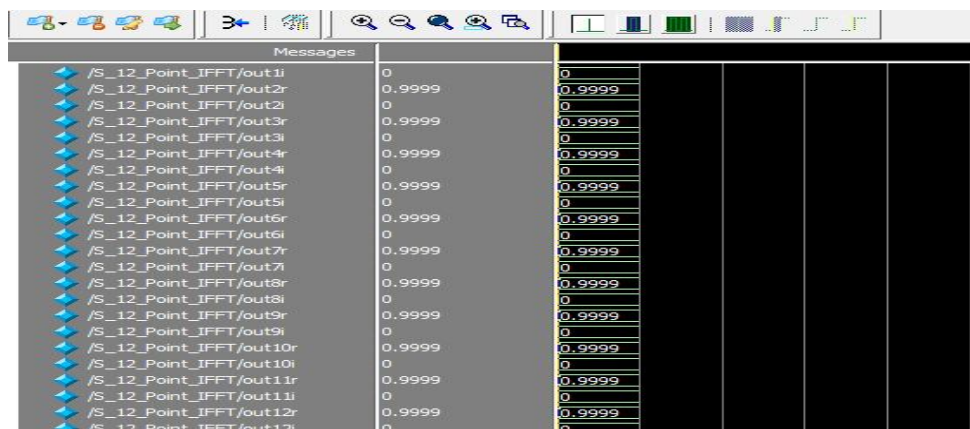


The image shows a screenshot of the Xilinx ISE Messages window. It displays the simulation results for the 12-point IFFT input sequence. The table below represents the data shown in the Messages window.

Message	Value	Value
/S_12_Point_IFFT/in1r	12	12
/S_12_Point_IFFT/in1i	0	0
/S_12_Point_IFFT/in2r	0	0
/S_12_Point_IFFT/in2i	0	0
/S_12_Point_IFFT/in3r	0	0
/S_12_Point_IFFT/in3i	0	0
/S_12_Point_IFFT/in4r	0	0
/S_12_Point_IFFT/in4i	0	0
/S_12_Point_IFFT/in5r	0	0
/S_12_Point_IFFT/in5i	0	0
/S_12_Point_IFFT/in6r	0	0
/S_12_Point_IFFT/in6i	0	0
/S_12_Point_IFFT/in7r	0	0
/S_12_Point_IFFT/in7i	0	0
/S_12_Point_IFFT/in8r	0	0
/S_12_Point_IFFT/in8i	0	0
/S_12_Point_IFFT/in9r	0	0
/S_12_Point_IFFT/in9i	0	0
/S_12_Point_IFFT/in10r	0	0
/S_12_Point_IFFT/in10i	0	0
/S_12_Point_IFFT/in11r	0	0
/S_12_Point_IFFT/in11i	0	0
/S_12_Point_IFFT/in12r	0	0
/S_12_Point_IFFT/in12i	0	0

Fig.7: IFFT 12-point sequence of Radix-3/6 algorithm

The Fig.7 Shows the simulation results of 12-Point Split Radix 3/6 IFFT input sequence $\{X_k\}=\{12,0,0,0, 0,0,0, 0,0,0, 0,0\}$ applied to the Modelsim for simulation.



The image shows a screenshot of the Xilinx ISE Messages window. It displays the simulation results for the 12-point IFFT output sequence. The table below represents the data shown in the Messages window.

Message	Value	Value
/S_12_Point_IFFT/out1i	0	0
/S_12_Point_IFFT/out2r	0.9999	0.9999
/S_12_Point_IFFT/out2i	0	0
/S_12_Point_IFFT/out3r	0.9999	0.9999
/S_12_Point_IFFT/out3i	0	0
/S_12_Point_IFFT/out4r	0.9999	0.9999
/S_12_Point_IFFT/out4i	0	0
/S_12_Point_IFFT/out5r	0.9999	0.9999
/S_12_Point_IFFT/out5i	0	0
/S_12_Point_IFFT/out6r	0.9999	0.9999
/S_12_Point_IFFT/out6i	0	0
/S_12_Point_IFFT/out7r	0.9999	0.9999
/S_12_Point_IFFT/out7i	0	0
/S_12_Point_IFFT/out8r	0.9999	0.9999
/S_12_Point_IFFT/out8i	0	0
/S_12_Point_IFFT/out9r	0.9999	0.9999
/S_12_Point_IFFT/out9i	0	0
/S_12_Point_IFFT/out10r	0.9999	0.9999
/S_12_Point_IFFT/out10i	0	0
/S_12_Point_IFFT/out11r	0.9999	0.9999
/S_12_Point_IFFT/out11i	0	0
/S_12_Point_IFFT/out12r	0.9999	0.9999
/S_12_Point_IFFT/out12i	0	0

Fig.8: IFFT 12-point sequence output



International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

The Fig.8 Shows the simulation results of 12-Point Split Radix 3/6 IFFT output sequence $\{x_n\} = \{0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999\}$.

V.FUTURE SCOPE

Implementation of 16 Point RADIX 3/6 FFT Design using Verilog and verification using system Verilog will be done. These implementations usually employ efficient fast Fourier transform (FFT) algorithms so much so that the terms "FFT" and "DFT" are often used interchangeably. The terminology is further blurred by the (now rare) synonym finite Fourier transform for the DFT, which apparently predates the term "fast Fourier transform" but has the same initialize.

VI.CONCLUSION

A radix 3/6 FFT algorithm is presented for length- 6^m DFT. The proposed algorithm is a mixture of radix-3 and radix-6 algorithm. It can evaluate a non-power-of-six DFT, as long as its length- 6^m can be divided by 6. In order to reduce the number of operations, all sub DFTs are reordered favourably. The proposed algorithm shows that its implementation requires less real operations as compared with the published algorithms. Its arithmetic complexity is about, which is close to that of standard SRFFT. Due to being an irregular integer for the sequence lengths, it is difficult to gain a completely accurate formula of computational complexity.

REFERENCES

1. M. Frigo and S. Johnson, "The design and implementation of fftw3," Proc. IEEE, vol. 93, no.2, pp. 216–231, 2005.
2. J. Keiner, S. Kunis, and D. Potts, "Using nfft 3—A software library for various nonequispaced fast Fourier transforms," ACM Trans. Math. Softw. (TOMS), vol. 36, no. 4, pp.19–19, 2005.
3. D. Sepiashvili, "Performance Models and Search Methods for Optimal FFT Implementations," M.Sc. thesis, Carnegie Mellon Univ., Pittsburgh, PA, 2000.
4. J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput, vol. 19, no. 90, pp. 297–301, 1965.
5. P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," Electron. Lett., vol. 20, pp. 14–6, Jan. 1984.
6. Kamar and Y. Elcherif, "Conjugate pair fast Fourier transform," Electron. Lett., vol. 25, no.5, pp. 324–325, Apr. 1989.
7. S. Bouguezel, M. Ahmad, and M. Swamy, "A new radix-2/8 FFT algorithm for length-DFTs," IEEE Trans. Circuits Syst. I, vol.51, no. 9, pp. 1723–1732, Sep. 2004.
8. E. Dubois and A. Venetsanopoulos, "A new algorithm for the radix-3 FFT," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-26, pp.222–225, Jun. 1978.
9. S. Prakash and V. Rao, "A new radix-6 FFT algorithm," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-29, no. 4, pp. 939–941, Aug. 1981.
10. Y. Suzuki, T. Sone, and K. Kido, "A new FFT algorithm of radix 3, 6, and 12," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-34, no. 2, pp. 380–383, Apr. 1986.