Implementation of Area Efficient 16point Radix 2^2 FFT Algorithm

P.Rajeshkumar*, K.Sai Tejaswini, D.Sushma Reddy, M.Akshita, A.Sarath Bala, N.Deepika

Department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad

Abstract:- The Discrete Fourier Transform (DFT) plays a significant role in many applications of digital signal processing. Basically, it has been applied in a wide range of fields such as linear filtering, spectrum analysis, digital video broadcasting and orthogonal frequency demodulation multiplexing (OFDM). The rapidly increasing demand of OFDM based applications, including modern wireless telecommunication such as LAN, needs real-time high-speed computation in Fast Fourier Transform algorithm. This has made the design of FFT processor a critical requirement for the upcoming wireless technology. With the advent of this requirement, the study of high performance VLSI FFT architecture is likewise of increasing importance. Many different hardware architectures have been proposed for the implementation of FFT algorithms. The main concern of the design approach will be power and architectural size. Among various FFT algorithms, radix-2 FFT with Cooley-Turkey algorithm, is very popular because it makes efficient use of symmetry and periodicity properties of the twiddle factor/coefficient

WnN = exp (-j2πn/N).Which reduce the computational complexity from $0(N^2)$ to $0(N\log_2 N)$.

Keywords: *FFT; Cooley-Turkey algorithm; radix-2^2; Butterfly structure.*

I. Introduction.

Several architectures have been proposed based on Cooley Turkey algorithm to further reduce the computation complexity, including radix-4, radix-2, and split-radix. Basically, this Fast Fourier Transform algorithm use Divide-and-Conquer approach to divide the computation recursively and then extract as many common twiddle factors as possible. The number of required real additions and multiplications is usually used to compare the efficiency of different FFT algorithms.

Structural regularity is also important in implementation of FFT algorithms on dedicated chips such as in ASIC (Application Specific Integrated Chip). Hence, radix-2 and radix-4 FFT algorithm are preferable in terms of speed and accuracy. This paper presents an area and power efficient 16-point radix-4 Fast Fourier Transform. The approach in re-utilizing the stored identical component enhances the physical fingerprint of the architecture. An improved complex multiplication is introduced in FFT butterfly computation to realize a cost efficient hardware. 16- point FFT radix-2^2 architecture is implemented utilizing 0.18µm technology from Artisan. The 16 bit imaginary and 16 bit real input-output is realized at 1.8V with operating frequency of 50MHz. The chip is designed for fixed-point data format. Great care had been taken into account to overcome the overflow issue in fixed-point data format. During the FFT computation, results at a particular stage are rounded and stored in the register memory. Since the FFT computation is an iterative process, the successive rounding errors at each output of butterfly accumulate over the FFT stages. The issue is solved by maintaining the error at the successive butterfly small. Twiddle factor/coefficient value are pre-calculated and stored in the register memory as 16-bit two's complement signed fixed-point words. The comparison results between conventional radix-4 and radix-2^2 architecture realized in 0.18µm CMOS technology are reported in simulation results. Fast Fourier Transform (FFT) has become almost ubiquitous and most important in high speed signal processing. Using this transform, signals can be moved to the frequency domain where filtering and correlation can be performed with fewer operations. It has been widely used in communications and radar applications. Recent advances in chip technology have increased field programmable gate array (FPGA) resources and as a consequence, the computation of complex algorithms, such as the FFT, can be implemented on a programmable device.

The real time requirements of FFT and the highly flexible FPGA form a great combination and improve the speed of FFT processing to meet the high-speed of the modern signal processing. There are various FFT algorithms such as radix-2, radix-4, radix2^2 , split-radix, wino grad and many more.Radix-2 algorithm is the simplest one, but its calculation of addition and multiplication is more than radix-4's.Though being more efficient than radix-2, radix-4 only can process 4n-point FFT. The most desirable hardware oriented algorithm will be that it has the same number of non-trivial multiplications at the same positions in the SFG as of radix-4 algorithms, but has the same butterfly structure as that of radix-2 algorithms. In this paper, a hardware-oriented radix-2^2 algorithm is compared with the conventional radix-2 algorithm, which has the radix-4 multiplicative complexity but retains radix-2 butterfly structure in the SFG.

II. Implementation of Radix 2^2 FFT.

These discrete Fourier Transforms can be implemented rapidly with the Fast Fourier Transform (FFT) algorithm

N	$(N-1)^2$	$(N/2)\log_2 N$
256	65,025	1,024
1,024	1,046529	5,120
4,096	16,769,025	24,576

Table.1 Comparison for various Sample values

FFTs are most efficient if the number of samples, N, is a power of 2. Some FFT software implementations require this.

Mathematica

Fourier[{$a_1,a_2,\ldots,a_N$}] $\quad \frac{1}{\sqrt{N}}\sum_{r=1}^{N} a_r \exp[2\pi i(r-1)(s-1)/N]$

↕ lists

InverseFourier[{$b_1,b_2,\ldots,b_N$}] $\quad \frac{1}{\sqrt{N}}\sum_{s=1}^{N} b_s \exp[-2\pi i(r-1)(s-1)/N]$

Maple

FFT(N, x_{re}, x_{im}) $\quad \sum_{j=0}^{N-1} x(j)\exp[-2\pi i jk/N]$

↕ arrays $\qquad N = 2^n$

iFFT(N,X_{re},X_{im}) $\quad \frac{1}{N}\sum_{k=0}^{N-1} X(k)\exp[2\pi i jk/N]$

MATLAB

fft(x) $\quad \sum_{j=1}^{N} x(j)\exp[-2\pi i(j-1)(k-1)/N]$

↕ arrays

ifft(X) $\quad \frac{1}{N}\sum_{j=1}^{N} X(j)\exp[2\pi i(j-1)(k-1)/N]$

A fast Fourier transform (FFT) is an algorithm that samples a signal over a period of time (or space) and divides it into its frequency components. These components are single sinusoidal oscillations at distinct frequencies each with their own amplitude and phase. Over the time period measured, the signal contains 3 distinct dominant frequencies.

Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805. In 1994, Gilbert Strang described the FFT as the most important numerical algorithm of our lifetime and it was included in Top 10 Algorithms of 20th Century by the IEEE journal computing in Science & Engineering. There are many different FFT algorithms based on a wide range of published theories, from simple complex-number arithmetic to group theory and number theory; this article gives an overview of the available techniques and some of their general properties, while the specific algorithms are described in subsidiary articles linked below.

The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields (see discrete Fourier transform for properties and applications of the transform) but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly: computing the DFT of N points in the native way, using the definition, takes $O(N^2)$ arithmetical operations, while an FFT can compute the same DFT in only $O(N \log N)$ operations. The difference in speed can be enormous, especially for long data sets where N may be in the thousands or millions. In practice, the computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to $N \log N$. This huge improvement made the calculation of the DFT practical; FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

The best-known FFT algorithms depend upon the factorization of N, but there are FFTs with O(N log N) complexity for all N, even for prime N. Many FFT algorithms only depend on the fact that is an N-th primitive root of unity, and thus can be applied to analogous transforms over any finite field, such as number-theoretic transforms. Since the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a $1/N$ factor, any FFT algorithm can easily be adapted for it. The algorithmic complexity of an FFT is usually quantified in terms of the total number of butterfly operations performed. Let this number be C (p) for a 2p point transform. Looking at the DIF routine above, it is easy to see that C(p) must satisfy the following recurrence relation. This has solution in terms of N (=2p).Dropping the constant scaling factors (including the log base) we get an algorithmic complexity of O(NlogN)

Types of FFT

- DIT-FFT(Decimation In Time-Fast Fourier Transform)
- DIF-FFT(Decimation In Frequency-Fast Fourier Transform)

III. Butterfly Structure.

In the context of fast Fourier transform algorithms, a butterfly is a portion of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa (breaking a larger DFT up into sub transforms). The name "butterfly" comes from the shape of the data-flow diagram in the radix-2 case, as described below. The earliest occurrence in print of the term is thought to be in a 1969 MIT technical report. The same structure can also be found in the Viterbi algorithm, used for finding the most likely sequence of hidden states.

Most commonly, the term "butterfly" appears in the context of the Cooley–Turkey FFT algorithm, which recursively breaks down a DFT of composite size $n = rm$ into r smaller transforms of size m where r is the "radix" of the transform. These smaller DFTs are then combined via size-r butterflies, which themselves are DFTs of size r (performed m times on corresponding outputs of the sub-transforms) pre-multiplied by roots of unity (known as twiddle factors).

A. DIT-FFT:

DIT algorithm is used to calculate the DFT of a N-point sequence.The idea is to break the N-point sequence into two sequences, the DFTs of which can be obtained to give the DFT of the original N-point sequence. Initially the N-point

sequence is divided into N/2-point sequences $x_e(n)$ and $x_0(n)$, which have even and odd numbers of x(n) respectively.

The N/2-point DFTs of these two sequences are evaluated and combined to give the N-point DFT.Similarly the N/2-point DFTs can be expressed as a combination of N/4-point DFTs.This process is continued until we are left with two point DFT.This algorithm is called decimation-in-time because the sequence x(n) is often split into smaller sequences.

B. DIF-FFT

In DIF-FFT algorithm, the output of DFT sequence is broken into smaller and smaller subsequences. The process of dividing the frequency components into even and odd parts is what gives this algorithm its name 'Decimation In Frequency'. If N is a regular power of 2, we can apply this method recursively until we get to the trivial 1 point transform. The factors TN are conventionally referred to as 'twiddle factors'.

C. Radix-2^2 DIF-FFT

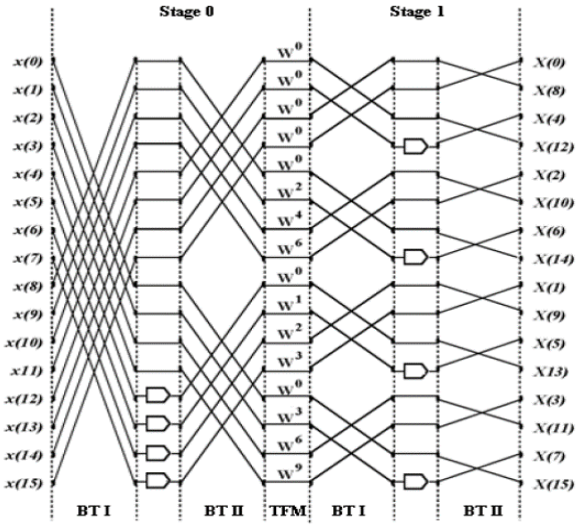

Fig1 Butterfly Structure of 16-point Radix 2^2 FFT

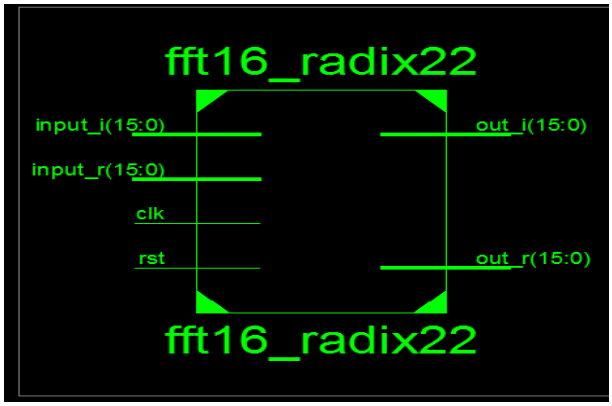

Fig2 Block Diagram of 16-point Radix 2^2 FFT

IV. Hardware Schematic

The Schematic of proposed architecture consists of following components:

1. Butterfly structure
2. Control unit
3. Delay Units
4. Processing Elements

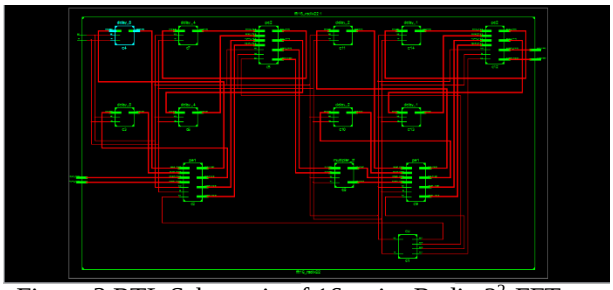

Figure 3 RTL Schematic of 16-point Radix 2^2 FFT

A. Butterfly structure

This section presents the identical radix-2^2 FFT algorithms, which are generated by using binary tree representation. Now based on the binary tree presentation of Cooley- Tukey FFT algorithm,. These algorithms can be generated by splitting N-point FFT into 2 k -point and $N - 2$ k -point. Apply this decomposition recursively until all sizes will become 2 k . This decomposition can apply in different ways to get identical radix-2 k FFT algorithms. Then apply radix-2 decomposition on all 2 k -point FFTs as shown in Fig1.

B. Control unit

Control Logic Unit: Control the implementation of the whole hardware operation that FFT/IFFT mode, FFT point, FFT stage are included. AGU: Controlled by Control Logic Unit, generates the read-address, write-address of memories and read-address of twiddle factor. Data Memory Group 1&2: Store raw data, intermediate processing data and result data. Twiddle Factor Unit: Store the twiddle factors for butterfly operation. Data Memory Group1&2 and Twiddle Factor Unit are included in Memory Unit. Butterfly Operation Unit: It contains two radix2 butterfly operation units and simultaneously reads 4 data for twice butterfly operations, then generates 4 results. Each unit works under the control of Logic Control U

C.Delay Unit

Delay units which provide delay of Delay1, Delay 2, Delay 4, and Delay 8 are used in this 16 point FFT.

D. Processing Elements

Processing elements processing element1 Processing element 2 are used in the two stage butterfly structure.

V. Results.

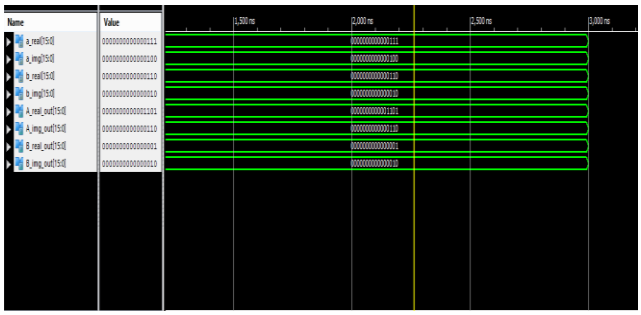

Fig 4 Simulation Output of Butterfly

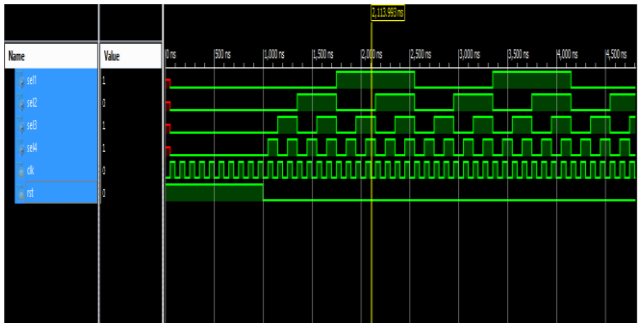

Fig 5 Simulation Output of Multiplier

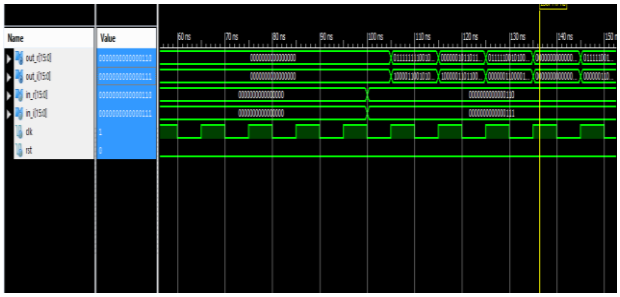

Fig 6 Simulation Output of Control Unit

Simulation results are observed using Xilinx version 10.1 and are shown in various figures in results section.

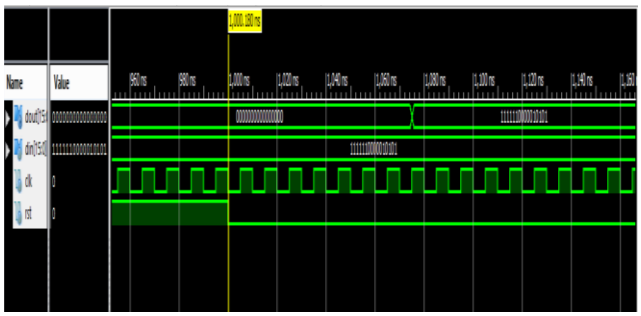

Fig 7 Simulation Output of Delay8

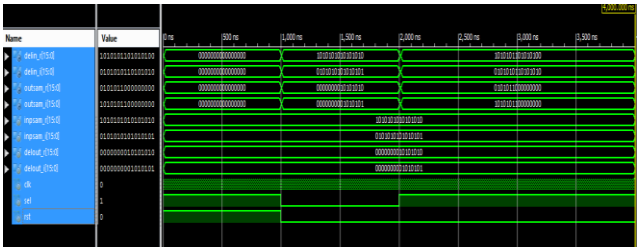

Fig 8 Simulation Output of Processing Element2

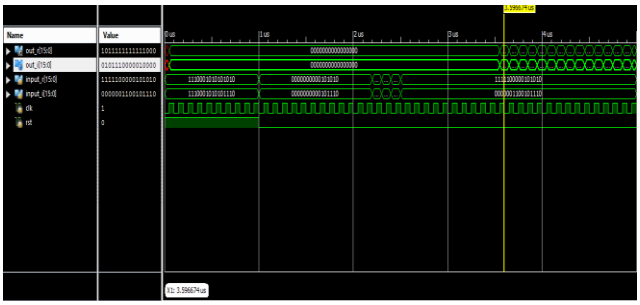

Fig 9 Simulation result of 16-point Radix 2^2 FFT

VI. Conclusion.

The paper introduces the design and implementation of a large points FFT acceleration unit in multi-processor system based on FPGA. The structure of hardware implementation is simple. It passes the software and hardware co verification as a part of the multi-processor image processing system. Results of FPGA verification are right within the range of allowable error. FPGA evaluation results prove that the design of this large point FFT acceleration unit is valid. With the development of FPGA technologies, it will become feasible for the implementation of large point FFT operation and make more progress in speed, volume and flexibility.

VII. References.

[1]. Anoop Thomas, Lakshmi Santhosh. "Implementation of Radix 2 and Radix 2^2FFT Algorithms on Spartan6 FPGA"(IEEE – 31661).

[2]. Anwar Bhasha Pattan, Dr. M. Madhavi Latha - "Fast Fourier Transform Architectures: A survey and state of the Art".

[3]. C. Sidney Burrus, Matteo Frigo, Steven G. Johnson, Markus Pueschel, Ivan Selesnick -"Fast Fourier Transforms".

[4]. FPGA implementation of a 64 point Radix-2 Single path Delay feedback FFT architecture by Anwar Bhasha Pattan, Makkena Madhavi Latha.

AUTHOR'S BIOGRAPHY

P.Rajeshkumar received B.Tech degree in Electronics & Communication Engineering from Bapatla Engineering College Bapatla, A.P, India and M.Tech degree in VLSI SYSTEMS DESIGN from JNTUK, Kakinada, A,P, India in 2004 and 2012 respectively.

Having 11 years of teaching experience currently he is working as Asst. Professor in the department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad, Telangana, India. His research interests include VLSI.

K.Sai Tejaswini is a Final year student of department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad, Telangana, India. She is a member of ISTE and IETE. Her area of interest includes Communication.

D.Sushma Reddy is a Final year student of department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad, and Telangana, India. She is a member of ISTE and IETE. Her area of interest includes Embedded Systems.

N.Deepika is a Final year student of department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad, and Telangana, India. She is a member of ISTE and IETE. Her area of interest includes Communication.

M.Akshita is a Final year student of department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad, and Telangana, India. She is a member of ISTE and IETE. Her area of interest includes Communication.

A.Sharath Bala is a Final year student of department of Electronics and Communication Engineering, BVRIT Hyderabad College of Engineering for Women, Hyderabad, Telangana, India. She is a member of ISTE and IETE. Her area of interest includes Communication.