International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Implementation of High Throughput Radix-16 FFT Algorithm

K. Swetha Sree[1], T. Lakshmi Narayana[2]

[1]Department of ECE, Andhra Loyola Institute of Engineering and Technology, Vijayawada, India

[2]Assistant Professor, Department of ECE, Andhra Loyola Institute of Engineering and Technology, Vijayawada, India

Abstract: ***The extension of radix-4 algorithm to radix-16 to achieve the high throughput of 2.59 giga-samples/s for WPAN's. We are also reformulating radix-16 algorithm to achieve low-complexity and low area cost and high performance. Radix -16 FFT is obtained by cascaded the radix -4 butterfly units. It facilitates low-complexity realization of radix-16 butterfly operation and high operation speed due to its optimized pipelined structure. The proposed radix-16 FFT algorithm is area-efficient with high data processing rate and hardware utilization efficiency. The multiplier cost of the proposed FFT algorithm is less than that of the previous FFT structures in FFT applications. Although the radix-16 FFT algorithm has less computational complexity, the control circuit of the direct radix-16 architecture for implementing radix-16 FFT is very complex. Thus, the efficient simplified radix-16 structure, with four radix 4 structures pipelined model, will be applied to radix-16 FFT algorithm, so that the extension is easily applicable.***

Keywords: Fast Fourier transforms (FFT), DFT, OFDM, radix-16 FFT, WPANs, pipelined structure

1. Introduction

Since the recent decade, the increasing demand for real time and high-rate multimedia services has been pushing the birth of high-rate wireless communication systems. Ultra wideband (UWB) communication system, for example, can deliver data rates up to 480 Mb/s at a short distance range. However, it is not enough to support high data rate applications of more than 1Gbps such as high-definition (HD) streaming content downloads HD video on demand, home theater, and etc. To meet the application demands, IEEE 802.15.3c-2009 standard for high-rate Wireless Personal Area Networks (WPANs) was ratified recently. In the standard, there are three PHY modes, Single Carrier mode (SC PHY), High Speed Interface mode (HSI PHY), and Audio/Visual mode (AV PHY). Except SC PHY, both HSI PHY and AV PHY are based on OFDM modulations. As is well known, Fast Fourier transform (FFT) operation is one of the key operations for OFDM-based communication systems. Besides, for SC PHY mode, FFT operations are widely employed for effective channel equalization, Wireless communication is becoming a must in electronic products .Since such products have distinct operation features, new wireless communication protocols, more suited for specific applications, are being devised. Currently, many contemporary electronic products adopt the IEEE 802.11n WLAN or the IEEE 802.16 WiMax wireless communication standards (or even both). Wireless modulation/demodulation requires the application of the Discrete Fourier Transform (DFT) and the Inverse Discrete Fourier Transform (IDFT). The high throughput can be given by using Block-floating point (BFP) operations are designed and implemented to achieve high signal-to-quantization-noise ratio (SQNR).

2. FFT Concept

2.1 The Discrete Fourier Transform (DFT)

The original DFT algorithm has an $O(n^2)$ complexity and, therefore, is not used in direct hardware realizations. The Discrete Fourier Transform (DFT) for an *N*-point sequence *x(n)* is given by , where *X(k)* and *x(n)* are complex numbers, *n* is the time index and *k* is the frequency index . By using this equation the DFT computation requires N^2 complex multiplications and $N(N–1)$ complex additions, leading to a complexity .

$$X(K) = \sum_{n=0}^{N-1} x(n).e^{-j2\pi kn/n} \text{.....eq 1}$$

A lower complexity DFT form, referred to as Fast Fourier Transform (FFT) algorithm, is used instead. The FFT algorithm has come into focus through a paper by Cooley and Tukey in 1965. Also, the Inverse Fast Fourier Transform (IFFT) is used to calculate the IDFT.

2.2 Basic FFT/IFFT Algorithms

Direct hardware implementations of the FFT algorithm either use the Decimation in Time (DIT) decomposition or the Decimation in Frequency (DIF) decomposition. In the DIF approach an *N*-point DFT is decomposed into two *N*/2-point DFTs, one for the even-indexed frequency outputs and another for the odd-indexed frequency outputs. The inputs to the even-indexed outputs *N*/2-point DFT are sums between first half inputs and second half inputs. The inputs to the odd-indexed outputs *N*/2-point DFT are differences between first half inputs and second half inputs, multiplied by constants that are referred to as "twiddle factors".

Each *N*/2-point DFT may be further decomposed into two *N*/4-point DFTs. Such decomposition may be applied recursively until reaching 2-point DFTs, which can be assumed as basic computation elements.

2.3Higher Radix

Benefit of higher-radix FFT operations over the lower-radix ones is that higher memory access bandwidth can be more conveniently provided in hardware implementations, as the bandwidth is proportional to. An improved radix-16 algorithm is proposed in which can reduce the numbers of twiddle factor operations and lookup-table accesses. For N-point FFT requires $\frac{N}{r}\log_r N$ FFT butterfly operations, higher-radix FFT algorithms can complete FFT operations with smaller numbers of FFT stages than the lower-radix algorithms. since generally a radix-16 butterfly unit is more complicated and less flexible than lower-radix ones, this work reformulates conventional radix-16 FFT algorithm so as to facilitate efficient and optimized pipelined realization of a radix-16 PE with high computing power and speed, high-performance radix-16 FFT processor which satisfies the mentioned throughput requirement of 802.15.3c applications.

3. 16 Point Radix 16 FFT Algorithms

3.1 Direct Radix 16 FFT:

This radix -16 FFT algorithm can be implemented directly using the DFT formula but the number of computations and additions will be increased using this formula and also when we want to implement this algorithm on the hardware like FPGA or the DSP processor the number of memory locations and the multiply- adder circuits increases which leads to the increase in the factors like power, implementation cost and area, the direct implementation will be as shown in the above figure

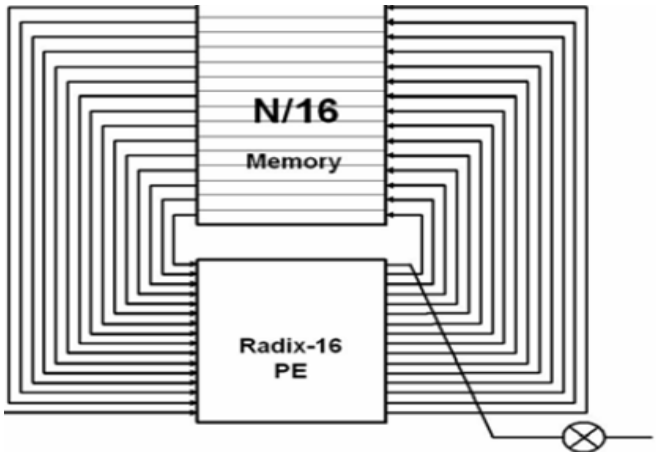

Figure 1: Direct implementation model of 16 radix 16 FFT using DFT points

The Discrete Fourier Transform (DFT) for an *N*-point sequence *x(n)* is given by , where *X(k)* and *x(n)* are complex numbers, *n* is the time index and *k* is the frequency index The equation used for the implementation is the equation 1. In that the x(n) sequence is divided in to even part and the odd part so number of multiplications require for one sample is N like that for n samples is N^2 similarly number of additions for one sample is N-1 for N samples is N(N-1) ,so for example if we take 16 samples we require 256 multiplications and 128 additions so when we implement it on the hardware the number of adder and multiplier circuits will increases to overcome this we are moving to the implementation of FFT algorithm.

3.2 FFT Algorithm Implementation:

The FFT algorithm is used to decompose N point the main aim of the FFT is to decompose the point and decreases the number of computations, this FFT algorithm is in two forms that is in DIT and DIF forms .In DIT form the number of the outputs are given in the form of N=r^M where m is a integer so if the N point sequence can be divided into N/r point where r is the radix so if the equation 1 is divided in to even index sequence and odd index sequence we will get N/r $\log_r N$ multiplications and N $\log_r N$ additions ,so when implemented over hardware we can reduce the number of multiplication units and memory locations , similar when we come to DIF FFT the number of input points can be expressed as N=r^M where M is number of stages.

Both algorithms require N $\log_r N$ operations to compute the DFT .Both algorithms can be done in –place and both need to perform bit reversal at some places during the computation

3.3 Radix 16 DIF FFT algorithm:

As the uses of the higher radix is explained in the above section we are moving to radix 16 algorithm to decompose the N point DFT by N/16 points .When compared to the other radix the radix 16 is useful to provide high throughput because the number of stages will decrees and for more number of sample we can implement in few number of steps and also when we implemented over hardware the number of multiplier and accumulators and adder units can be reduced .But even radix 16 is less complex to design it using signal flow graph is very complex so we are moving to it reformulated structure.

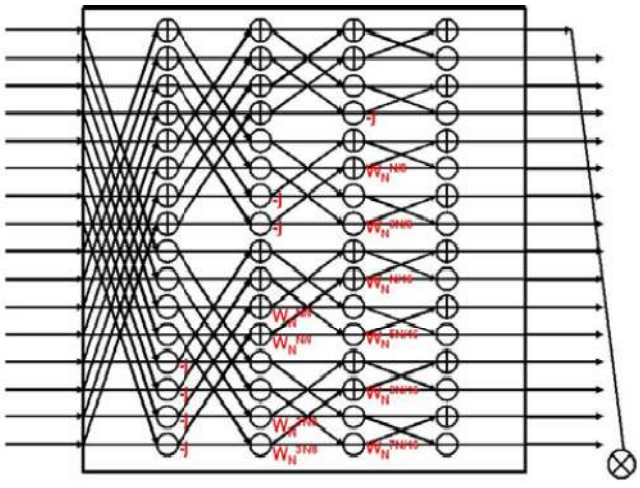

Figure 2: 16 point Radix 16 block structure

3.4 Reformulated 16- Point Radix 16

As we are design for 16 point radix-16 in DIF FFT model the number of inputs will be 16 and number of stages will be one know we will replace that stage and replace it with four radix 4 units where the input for the each unit is 4 samples that will in the form

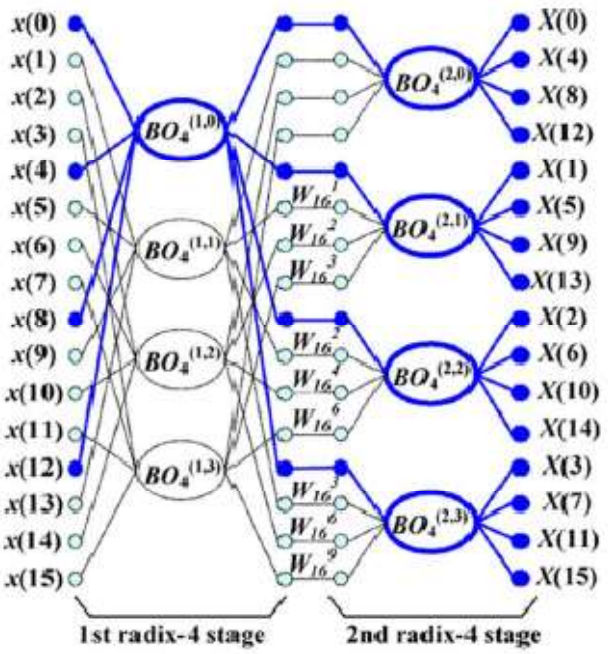

Figure 3: Reformulated 16 point radix 16 structure

So for N point DFT is given from the above formula (eq 1) so here we have taken four radix 4 units and we have taken m1, m2, m3,m4 as the sub stages we have divided the samples from 0 to 3, 4 to 7 and we have taken l1, l2 ,l3 are the sub stages so we have used the pipelined instruction set here as explained in the following formula

$$B\ O_{16}^{(1,n_2)}(k_1) = \sum_{m_1=0}^{3}\sum_{m_2=0}^{3} x(16(4m_1+m_2)+n_1)w_4^{(4m_1+m_2)})w_{16} \ldots\ldots\ldots\ldots\text{eq 2}$$

In each stage simpler cascaded radix-4 butterfly sub-stages than the original radix-16 butterfly stage, because the coefficients of 4-point DFT are all trivial computations .Consequently, a radix-16 butterfly can be efficiently executed in a four-cycle period. During the period, the partial results of all 16 output samples will be pipelined simultaneously.

3.5 Extension of16 point radix 16 FFT algorithm

The 16 point radix 16 FFT algorithm is extended to 32 point radix 16 in this we have decomposed FFT to two N/16 stages so from this extension we can prove that the number sample point increases we can decrees the computing factor and also hardware implementation increase the speed by the formula

$$\text{Speed improvement factor} = \frac{N^r}{\frac{N}{r}\log_r N} \ldots\ldots\ldots\ldots\ldots\ldots\text{eq 3}$$

4. Simulated Results

The 16 point radix 16 is implemented in mat lab , to compute this we have design the function for radix 16 and showed that number samples given to each stage can be increased and calculated through put using these two functions

32 point radix 16 output:

Columns 1 through 16
10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i 10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i
10.0000 -2.0000 - 2.0000i 2.0000 2.0000 - 2.0000i 10.0000 - 2.0000 + 2.0000i -2.0000 2.000-2.0000i

Columns 17 through 32
74.0000 -8.0000 + 8.0000i -14.0000 -8.0000 - 8.0000i 44.0000 -8.0000+8.0000i -14.0000 -8.0000-8.0000i
-8.0000 + 8.0000i -14.0000 -8.0000 - 8.0000i 41.0000 - 8.0000 + 8.0000i 14.0000 -8.0000 + 8.0000i -14.000

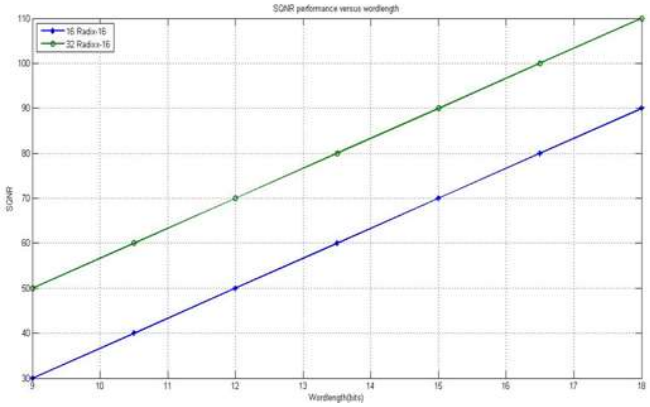

Figure 4: Through between 16 and 32 point radix 16

5. Conclusion

In present DSP area it is one of the important issues that to be addressed. this proposed method is implemented in mat lab .simulated are targeted to decreases the number of computations and increases the throughput by increasing the radix in FFT for better decomposing on N point DFT .Thus this proposed method is more suitable for reduction of hardware components in DSP processor

References

[1] IEEE 802.15.3c-2009: Part 15.3: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*.
[2] S. Johansson, S. He, and P. Nilsson, "Word length optimization of a pipelined FFT processor," in *Proc. 42ndMidwest Symp. Circuits Syst.*., 1999, pp. 501–503.
[3] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *Proc. URSI Int. Symp. Signals, Syst., Electron,* 1998, pp. 257–262.
[4] Y. N. Chang and K. Parhi, "An efficient pipelined FFT architecture," *IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process.* vol. 50, no. 6, pp. 322–325, Jun. 2003.
[5] D. Cohen, "Simplified control scheme of FFT hardware," *IEEE Trans. Signal Process.*, vol. ASSP-24, no. 12, pp. 577–579, Dec. 1976.
[6] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," *IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process.* vol. 39, no. 5, pp. 312–316, May 1992.