# An Efficient Fused Arithmetic Model for FFT in OFDM applications

Sharmila Hemanandh[1], A. Sivasubramanian[2]
[1]*Department of Electronics and Communication Engineering, Sathyabama University, Chennai, India*
[2]*Department of Electronics and Communication Engineering, VIT University, Chennai, India.*

**Abstract - This investigation proposes the radix 2, radix 4 and radix 8 FFT algorithms with improvement in performance using fused floating point arithmetic units. Multiplication operation dominates the execution time and hardware circuitry required in implementing the FFT algorithms. This study focuses on the identification of redundant computation in both FP and FFT modules based on hardware fusion technique. The Distributed Arithmetic (DA) based Canonical Signed Digit (CSD) technique is applied for suitable mantissa computation. Moreover higher radix indices with improved system performance are presented without adding any pipelining and parallel processing techniques. The comprehensive comparison results indicate that the proposed algorithms exhibits reduced area and high operating frequency, and thus have the highest efficiency.**
**Keywords: Fast Fourier transform, Distributed Arithmetic, Canonical Signed Digit, Floating Point.**

## I. INTRODUCTION

Discrete Fourier Transform (DFT) is one of the most important tool used in the field of signal processing for a wide variety of modern signal processing applications. DFT plays a vital role to analyze the spectral characteristics of a signal in various digital signal processing applications such as image processing, OFDM, digital terrestrial television broadcasting, spectral estimation, voice recognition, and other filtering applications. Fast Fourier transform(FFT) is the most competent algorithm used to compute the DFT using Decimation in Time and Decimation in Frequency algorithms. High speed data transmission over multipath channel environment with reduced symbol interference is achieved using OFDM[1]. Hence OFDM commands highly accurate arithmetic model to overcome the synchronization errors in both time and frequency domain.

Major challenges involved in OFDM system using high precision computation can be the use of floating point arithmetic that involves more complex arithmetic and logical operations[2]. Each of the functional units can be optimized to support low complexity and energy efficiency. Secondly FFT optimization and hardware fusion includes both area and speed constrains of the FFT butterfly unit that can be optimized by FFT factorization of the twiddle factors. Using hardware fusion reduced arithmetic redundancy is achieved in FFT computation. Previous research majorly concentrated on factors that include the use of floating point units for efficient FFT computation, Optimization of floating point arithmetic model and the use of fused arithmetic model to reduce hardwarecomplexity. A survey of these various factors show that hardware optimization through fusion approach is used as a simplest process to accomplish low complexity.

Yunhua Wang et al. (2007)[14] proposed a novel real time algorithm to convert 2's complement number to CSD using few combinational logic gates. The performance of the iterative multiplier is further improved by introducing a radix 8 hardware support. The iterative multiplier delivers significant improvement in performance with respect to speed area and power.

Hani H. Saleh et al. (2008)[7] proposed floating point fused dot product unit that performs floating point multiplication followed by addition on a pair of data. The fused dot product unit is 27 percent faster and more accurate than the conventional approach.

Arioua and Hassani (2014)[5] used CSD representation to save the twiddle factors in order to implement the complex multiplier. Thus the area and power consumption of the complex multiplier is reduced. A low arithmetic complexity FFT/IFFT processor is designed and developed for high speed low power OFDM based wireless communication system.

Earl E. Swartz landerJr and Hani H.M. Saleh (2012)[6] proposed two fused units namely two term dot product unit and add sub unit. The fused units are used in the implementation of radix 2 and radix 4 butterfly elements. The fused FFT butterflies are 15 percent faster and 30 percent smaller than the conventional implementation.

Prasanna Palsodkar and Ajay Gurjar (2014)[11] proposed designed two fused floating point primitives to speed up DSP hardware with respect to IEEE754 single precision format that supports all rounding modes.

Prasanna Palsodkar and Ajay Gurjar (2016) [12]analyses the implementation of radix 2 Decimation in Frequency butterfly unit using the fused units. By the use of fused primitives area is reduced by 27.09% and consumes 11% less power when compared to the discrete implementation.

Amir Kaivani and Seok-Bum Ko (2015)[3] proposed a floating point fused dot product add unit to improve the performance of FFT butterfly unit. The proposed design consumes low area at the cost of latency overhead.
Amir Kaivani and SeokbumKo (2015)[4] uses redundant floating point number system in the design of high speed butterfly architecture. The redundant number representation of the significant and the proposed floating point fused dot product add unit contributes to the improvement in speed at the cost of higher area.

## II. FLOATING POINT FUSED UNITS

The methodology used in IEEE 754 floating point arithmetic units for mantissa computation has major impact that leads to obvious performance merits with regard to low cost hardware requirements. In many existing works fused floating point units are employed to increase the execution speed of the butterfly operation. Fused units play a major role in many DSP applications. Implementation of FFT algorithm using fused unit shows considerable reduction in power and area when compared with discrete floating point adder followed by multiplier.
The performance of floating point arithmetic is improved by using fused floating point units when compared with the standard floating point arithmetic units. Fused units play a major role in the implementation of many DSP algorithms with increased speed, reduced latency, and hardware cost. Two fused floating point units namely Fused add sub unit (FAS) and Fused dot product unit are used in the design of higher radix FFT algorithm.

### *2.1. Floating Point Fused Add Sub Unit*

The FAS unit computes the sum(X+Y) and the difference(X-Y) simultaneously. To begin with the exponent compare logic compares the exponents of the two operands to identify the smaller number and also computes the difference between the two exponents.The exponents of the two operands must be made equal before addition or subtraction. If the exponent of X is greater than that of Y, then right shift the mantissa of Y by the difference value of the two exponents.Similarly if the exponent of Y is greater than that of X, then right shift the mantissa of X by the difference value of the two exponents. By doing so the exponents of the two operands are made equal. Next the sum of the two operands is computed. Normalization is required if the results lie outside the permitted range. If the result obtained after adding the two operands is not in the prescribed format, then it has to be rounded. The fused add sub unit is designed to handle all the four special values and all rounding modes specified by IEEE754 standard.

### *2.2. Floating Point Fused Dot Product (FDP) Unit*

The sum and the difference of the products of two operands is required in the computation of FFT. FDP unit improves the performance of many DSP algorithms. This unit is of great importance in the multiplication of two complex numbers.
Let A = are+ aim and B = bre + bim be two complex numbers. Then the product Y = AB is derived as
$Y = (are + aim) \times (bre + bim) = (arebre – aimbim) + j(arebim + bre aim)$
The above equation requires two adders and four multipliers when implemented with discrete floating point adders and multipliers. Alternatively only two fused dot product units are required to implement the equation.
When compared to the conventional dot product unit the fused dot product unit exhibitsconsiderable reduction in area when compared to the discrete parallel implementation of two multipliers and an adder since a single set of hardware for exponent adjust and significand shift are shared by both add and subtract operation.
The results obtained are more accurate because only one rounding operation is used instead of three for the conventional approach. The conventional FDP unit employs a tree based multiplier to multiply the mantissa of the two operands. This research proposes a multiplier based on CSD format. The proposed CSD based multiplier increases the possibility of zero partial product to 66.7%. This in turn reduces the power consumed by the multiplier. Hence the performance of the FDP is improved when compared to the conventional FDP unit, which in turn improves the overall performance of the FFT. Radix 2, radix 4 and radix 8 FFT algorithms are implemented using the proposed multiplier.

### *2.3. CSD Number Representation*

Multiplication is an important arithmetic operation involved in DSP applications. Nowadays, all modern engineering and technology applications prefer floating point in the field of signal processing for scientific computations due to its high dynamic range when compared with fixed point representation. Hence it is essential to improve the speed of the floating point multiplier. Conventional array multipliers and parallel are used for high speed multiplication at the expense of large area. Multiplication involves two steps namely partial product generation and partial product accumulation. The performance of the multiplier can be improved by reducing the number of partial products generated and by accelerating the accumulation of partial products. The complexity of the multiplier can be reduced with smaller number of partial products which in turn reduces the time needed to accumulate the partial products.

Canonical Signed digit recoding is a technique used to reduce the number of partial products. CSD representation has two main properties 1. Minimum number of non zero digits which results in reduced number of additions. 2. No two consecutive digits are non zero which facilitates multiple representation for a single binary number.
The basic functional units of a multiplier with CSD representation are CSD recoding unit, 2's complement unit, shift control unit and adder unit as shown in Figure1.

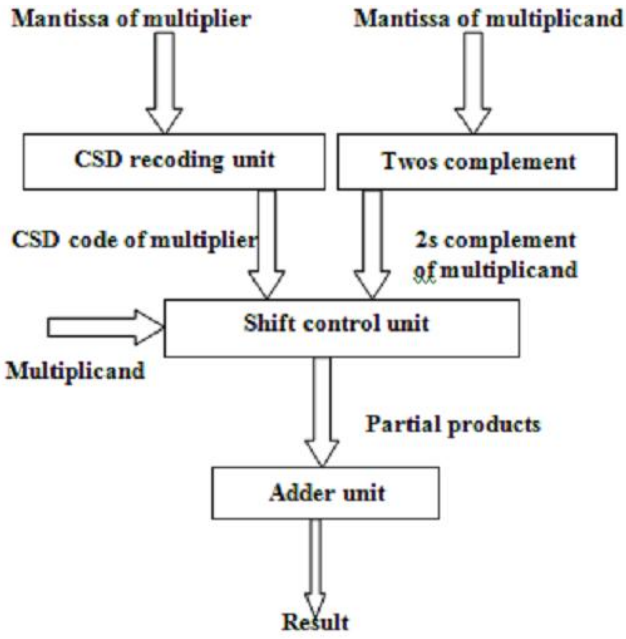


Figure 1. Floating point multiplier using CSD

The algorithm to multiply two floating point numbers with CSD representation is
The mantissa of the multiplicand is fed to the 2's complement unit and the mantissa of the multiplier is fed to the CSD recoding unit.
The 2's complement of the multiplicand and the CSD code of the multiplier are fed to the shift control unit. Partial products are generated by shifting. The number of shifts are based on the sign and position of the non zero digit.
The partial products are then added in the adder unit.
The results are rounded and then normalized to get the final product.

*2.4. Fast Fourier Transform (FFT)*
Fast Fourier transform is an efficient algorithm used to compute the DFT using Decimation in Time and Decimation in Frequency algorithms. The basic building block of the Fast Fourier transform algorithm is the butterfly structure. Figures 3.9 and 3.10 show the basic butterfly structures that define the decimation in time and decimation in frequency algorithms respectively. Both the algorithms exhibit the same computational complexity but differ in the arrangement of input and output.

*2.5. FFT Factorization*
The most popular radix 2 method of computing the FFT algorithm was proposed by Cooley and Turkey. The radix 2 decimation in time and decimation in frequency algorithms is the simplest FFT algorithms. In radix 2 algorithm the length of the sequence is always expressed in powers of 2. The basic operations involved in radix 2 DIF FFT butterfly are addition and multiplication. Based on divide and conquer approach many algorithms such as radix 4, radix 8, and split radix are developed to reduce the computational complexity.

*2.6. Radix 4 FFT Algorithm*
The computational complexity of radix 2 algorithms can be reduced further by using radix 4 algorithm. Radix 4 algorithm exploits the fact that the multiplication of the input with the twiddle factors especially +j and –j is carried out without the use of a complex multiplier. In radix 4 algorithm the N point DFT is divided into four N/4 point DFT. The basic butterfly diagram of radix 4 decimation in frequency FFT is shown in Figure 3.11. The radix 4

butterfly has 4 inputs x(n), x(n+N/4), x(n + N/2), and x(n + 3N/4). The N point DFT is computed as the sum of the outputs of the four N/4 point DFTs. The four N/4 point DFTs together represent the N point DFT. Radix 4 algorithm requires only 75% of the complex multiplication utilized in the computation of radix 2 algorithm.

*2.7. Radix 8 Algorithm*

Radix 8 algorithms further reduce the computational complexity of radix 2 and radix 4 algorithms. The basic butterfly diagram of radix 8 algorithm has 8 inputs. The radix 8 algorithm rearranges the N point DFT into eight N/8 point DFTs. The output of the N/8 point FFT is reused which greatly reduces the computational complexity of the algorithm. The output of the butterfly is computed as the sum of the outputs of all the N/8 point DFTs. Figure 3.4 shows the butterfly diagram of radix 8 algorithm.

*2.8. Floating Point FFT*

An efficient floating point arithmetic model for FFT computation should provide full support in reducing the hardware complexity and the richness in operating frequency. Bridging the performance gap between fixed point and floating point model is a big challenge. Though various research has been undertaken, floating point systems have failed to perform satisfactorily both in terms of speed and energy efficiency. The system is also to focus on parameter retention of the proposed floating point model over various high speed radix FFT computations.

The generic performance metrics of OFDM system includes both the estimation techniques used and arithmetic model used for FFT computation. Since floating point models are exploiting higher dynamic ranges and includes fraction parts for twiddle factors it is capable of providing significant error reduction with any linear MMSE estimator. Whereas the fixed point model is not adequate for mitigating the channel offset values over higher order modulation techniques.

Here complexity reduction is achieved with significant improvements in execution speed. The use of fused model gives unified dot product results required for basic FFT butterfly computation which is used as basic computational kernel

The results obtained are more accurate because only one rounding operation is used instead of three for the conventional approach. The conventional FDP unit employs a tree based multiplier to multiply the mantissa of the two operands. This research proposes a multiplier based on CSD format. The proposed CSD based multiplier increases the possibility of zero partial products to 66.7%. This in turn reduces the power consumed by the multiplier. Hence the performance of the FDP is improved when compared with the conventional FDP unit,which in turn improves the overall performance of the FFT. Radix 2, radix 4 and radix 8 FFT algorithms are implemented using the proposed multiplier. The results for various FFT radix point are presented to demonstrate the performance of the approaches.

## III. RESULTS AND DISCUSSION

This work analyses the trade off measures of various FFT radix implementations using different FP models. The proposed CSD based floating point model is implemented using the Verilog HDL and synthesized using ALTERA Cyclone III EP3C16F484C6 device. The results obtained are tabulated to validate the area and power metrics.

From the results it is proved that the use of CSD with bit normalization is 5% area efficient over FFT with Fused model and offers 6% reduction in hardware complexity as shown in table 1.

*3.1 Radix-4 DIF FFT*

From results it is proved that the use of CSD with bit normalization is 6% area efficient over FFT with Fused model and offers 14% reduction in hardware complexity as shown in table 2.

Table 1. Performance comparison of CSD over fused method using Radix-2 DIF FFT Butterfly Unit.

| Method used for floating point arithmetic | AREA (LE's used) | Power (mW) | Speed (MHz) |
|---|---|---|---|
| Fused | 6807 | 199.58 | 70.19 |
| CSD | 6454 | 182.97 | 73.46 |

Table 2. Performance comparison of CSD over fused method using Radix-4 DIF FFT Butterfly Unit

| Method used for floating point arithmetic | AREA (LE's used) | Power (mW) | Speed (MHz) |
|---|---|---|---|
| Fused | 4340 | 94.56 | 74.56 |
| CSD | 4091 | 92.44 | 73.8 |

*3.2 Radix-8 DIF FFT*

From results it is proved that the use of CSD with bit normalization is 10% area efficient over FFT with Fused model and offers 16% reduction in hardware complexity as shown in table 3.

Table 3. Performance comparison of CSD over fused method using Radix-8 DIF FFT Butterfly Unit.

| Method used for floating point arithmetic | AREA (LE's used) | Power (mW) | Speed (MHz) |
|---|---|---|---|
| Fused | 3405 | 93.01 | 74.76 |
| CSD | 3045 | 88.77 | 74.93 |

## IV. CONCLUSION

A fused floating point model based Fast Fourier transform with respect to radix-2/4/8 algorithm is proposed. The efficiency of selective CSD modeling in twiddle factor multiplication gave considerable delay reduction in all the three types FFT radix algorithms.. Based on the evaluation results simple hardware efficient FP FFT core model was structured. The results indicate that the CSD driven FP FFT model is able to support high speed computation more efficiently and also achieve considerable computational complexity reduction.

## V. REFERENCES


[1] Ghassemi, A. and Gulliver, T. A. (2007) Finite Word length Design for FFT/IFFT in UWB-OFDM Systems", Wireless Telecommunications Symposium, pp. 1 – 7.

[2] Hani Saleh Earl E. Swartzlander, Jr (2008), "A Floating-Point Fused Add-Subtract Unit", 51st Midwest Symposium on Circuits and Systems, pp. 519-522.

[3] Amir Kaivani and Seok-Bum Ko (2015), "Area efficient floating-point FFT butterfly architectures based on multi-operand adders", Electronics Letters, Vol. 51, No. 12, pp. 895-897.

[4] Amir Kaivani and SeokbumKo (2016), "Floating-Point Butterfly Architecture Based on Binary Signed-Digit Representation". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 24, No. 3, pp.1208-1211.

[5] Arioua, M. and Hassani, M. M. (2014), "A Low Multiplicative Complexity of Proposed FFT/IFFTs Design Applied for OFDM-Based Wireless Communication Systems", Journal of Advances in Computer Networks, Vol. 2, No. 1, pp. 24-27.

[6] Earl E. SwartzlanderJr and Hani H.M. Saleh (2012), "FFT Implementation with Fused Floating - Point Operations", IEEE Transactions on Computers, Vol. 61, No. 2, pp. 284-288.

[7] Hani H. Saleh Earl E. SwartzlanderJr (2008), "A Floating-Point Fused Dot-Product Unit", IEEE 21st Symposium on Computer Arithmetic pp.427-431.

[8] Huang, L., Shen, L., Dai, K. and Z. Wang (2007), "A new architecture for multiple-precision floating point multiply-add fused unit design", in Proc. 18th IEEE Symp. Comput. Arith., pp. 69-76.

[9] Liang, S., Tessier, R. and Mencer, O. (2003), "Floating point unit generation and evaluation for FPGAs", in Proc. 11th Annu. IEEE Symp. Field-Program. Custom Comput. Mach., pp. 185-194.

[10] Liou, C. and Chiueh, H. (2008), "An ALU cluster with floating point unit for media streaming architecture with homogeneous processor cores", in Proc. IEEE 13th Asia-Pacific Comput. Syst. Arch. Conf., pp. 1–7.

[11] PrasannaPalsodkar Ajay Gurjar (2014), "Improved Fused Floating Point Add-Subtract Unit for FFT Implementation", International Conference on Electronics and Communication Systems (ICECS), pp 1-5.

[12] Prasanna Palsodkar1 and Ajay Gurjar (2016), "Fused Floating Point Arithmetic Unit for Radix 2 FFT Implementation", IOSR Journal of VLSI and Signal Processing, Vol.6, No.2, pp.58-65.

[13] Ruiz, G.A. and Manzano, M.A. (2001), "Self-timed multiplier based on canonical signed-digit recoding", IEEE Proceedings – Circuits Devices Systems, Vol. 148, pp 235-241.

[14] Yunhua Wang (2007), "Iterative Radix-8 Multiplier Structure Based on a Novel Real-time CSD Recoding Signals", Systems and Computers, 2007, ACSSC 2007, pp. 977-981.