The Report committee for Yeong Foong Choo

Certifies that this is the approved version of the following report:

Complex Block Floating-Point Format with Box Encoding in Communication Systems

APPROVED BY

SUPERVISING COMMITTEE:

Brian L. Evans, Supervisor

Earl E. Swartzlander Jr.

Complex Block Floating-Point Format with Box Encoding in Communication Systems

by

Yeong Foong Choo

Report

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

The University of Texas at Austin
May 2018

This report is dedicated to my wife, Mun and my newborn daughter, Grace.

Acknowledgments

I would like to take this opportunity to thank Professor Brian L. Evans for allowing me to conduct this research topic in communication systems design. I am also thankful for the idea generation from Dr. Alan Gatherer at Huawei Labs. With the guidance of Dr. Gatherer and Dr. Evans, I had the opportunity of presenting my first publication at 2017 Asilomar Conference on Signals, Systems, and Computers. I would also like to offer my appreciation to Professor Earl E. Swartzlander Jr. for his willingness to be the reader of this report and for his detailed attention in suggesting improvement on this report.

My experience in pursuing the graduate degree of Electrical and Computer Engineering (ECE) is enriched by participating in all research-oriented networking events within the Embedded Signal Processing Lab (ESPL) and the Wireless Networking Communications Group (WNCG). The feedback that I received from Jinseok Choi, Junmo Sung, Faris Mismar, Matthew Koenig, and other remote ESPL members over the duration of my research project have been very valuable and ensuring the quality of my research presentation. At the Texas Wireless Summit 2017 and WNCG Open House 2018, Dr. Amine Mezghani was a consistent visitor at my poster presentation who offered valuable feedback at potential improvement of this project.

Complex Block Floating-Point Format with Box Encoding in Communication Systems

by

Yeong Foong Choo, MSE
The University of Texas at Austin, 2018
Supervisor: Brian L. Evans

This research project entails an efficient numeric digital representation in communication systems design. A complex block floating-point format with box encoding is proposed to encode an array of complex numbers that has better numeric resolution than its IEEE-754 counterpart when the same number of bits are allocated to the dominant value in the array. It is estimated that at least 10% of bit savings could be achieved by the new complex block representation on a quad-precision IEEE-754 format. A further bits savings of up to 18% could potentially be achieved for complex blocks at half-precision and single-precision IEEE-754 representation.

The implementation cost of the proposed block floating-point format is evaluated in terms of memory usage, design of arithmetic units, and memory input/output rates for communications system modeling and block diagrams. Further analysis is performed on the limitation and quantization effects of this complex block format relative to complex IEEE-754 format. The coverage of the arithmetic units design include complex block adder and complex block multiplier. The appropriate systems that would be required to perform algorithms such as the fast Fourier transform (forward and inverse) are designed using the proposed complex block format in multi-stages complex block multiply-adder.

The proposed block floating-point format is simulated as a new numeric class defined and implemented in MATLAB simulation environment. The MATLAB simulation is divided into two major parts. The first part of MATLAB simulation targets the simulation of complex block addition and complex block multiplication units for arbitrary size of complex samples per input block. The reference output values of complex block arithmetic are those computed with similar precision in IEEE-754 format. The second part of MATLAB simulation is performed on the system model of the single-carrier modulation-based and multi-carrier modulation-based communication systems. The quadrature amplitude modulation (QAM) is the baseband modulation type targeted in this work. The specification identified in the system model is relevant to those specified in the Long-Term Evolution (LTE) Standards for Base Station, Release 12.

Table of Contents

List of Tables

List of Figures

Chapter 1

Implementation of Digital Communication Systems

1.1 Digital Communication Systems Brief

The digital communication systems take various forms. The basic unit of information is a logical bit of a "0" or "1" and the rate of information transfer is measured in terms of bits/s. Multiple fixed-length bits are grouped into a symbol. Each symbol is mapped to a unique symbol amplitude and the symbol amplitudes are converted into a continuous-time analog baseband (low frequency) signal. The baseband signal is then upconverted to a carrier frequency for bandpass transmission. Figure 1.1 shows the block diagram for simple communication systems with abstraction on transmitter baseband processing (TX BB), upconversion unit, channel model, downconversion unit, and receiver baseband processing (RX BB).

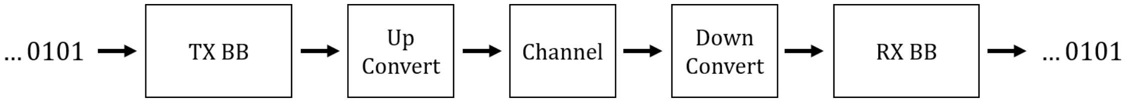


Figure 1.1: Block Diagram of Simple Communication Systems

Any communication system would be restricted to occupying finite bandwidth for information transfer. The modulating signal has its fundamental frequency at the center of the transmission frequency band. Baseband modulation allows one-dimension baseband symbol be modulated on the modulating signal. Amplitude modulation allows information be encoded as symbols with varying amplitude. Phase shift keying allows symbols to be encoded in the phase of a carrier signal. The combination of amplitude and phase modulation is a common way to use the transmission frequency band efficiently. The combination of amplitude and phase modulation leads to the study of complex baseband modulation or quadrature amplitude modulation (QAM).

Figure 1.2 shows the block diagram of transmitter baseband signal processing for quadrature amplitude modulation (QAM). QAM is preferred for high data rates since it effectively increases the data rate by 2x for the same transmission bandwidth and signal-to-noise ratio (SNR) condition. Information is carried independently on two parallel streams of amplitude symbols known as In-phase and Quadrature-phase signals. Both In-phase and Quadrature-phase signals share a common oscillator circuitry and therefore sharing the same transmission band centered at the intermediate frequency, f_{im}, but the Quadrature-phase signals have phase offset of $\frac{\pi}{2}$ relative to the In-phase signals. The pair of In-phase and Quadrature-phase signals is commonly known as complex IQ signals.

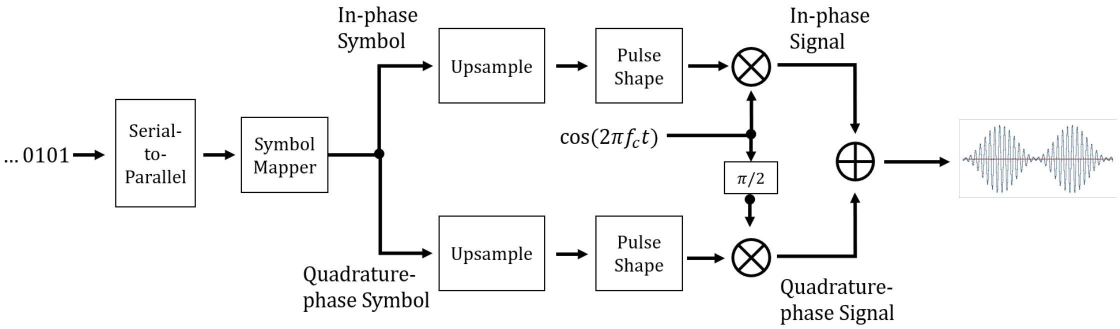


Figure 1.2: Block Diagram of Transmitter Baseband Processing

Figure 1.3 shows the block diagram of receiver baseband signal processing for quadrature amplitude modulation (QAM). The channel effect introduces signal distortion and degradation to the received signals and requires additional recovery work performed at the receiver. Often, the receiver would perform higher than Nyquist sampling rate to provide additional samples for running phase and timing offset correction algorithm. Furthermore, the channel equalization algorithm requires that the received data samples be available in a block for processing and higher throughput.

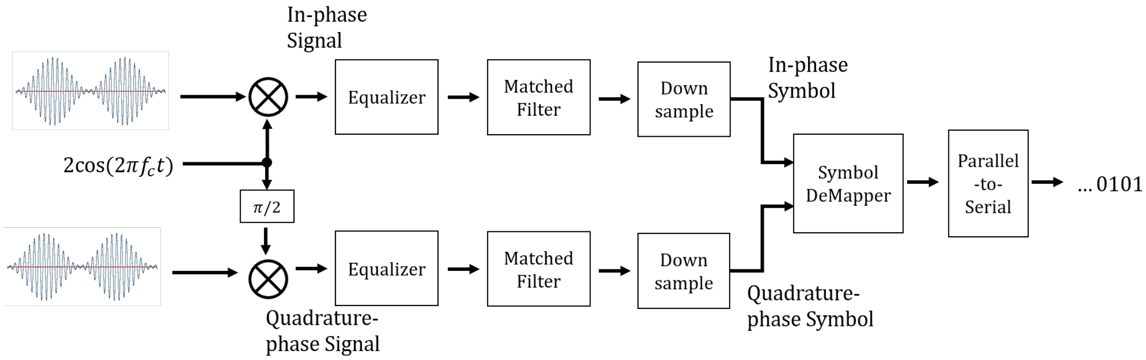


Figure 1.3: Block Diagram of Receiver Baseband Processing

The channel behavior determines the selection of modulation type. Single-carrier modulation refers to shifting the baseband (low frequency) signal to be centered at a carrier frequency, f_c. Multi-carrier modulation refers to the composition of several narrowband signals in the baseband frequency region that each of them carries baseband symbol (Narrowband means that the transmission bandwidth is much smaller than the carrier frequency). More details on transmitter and receiver design in signal-carrier modulation and multi-carrier modulation are described in Section 4.1 and Section 4.2. For frequency flat fading channels, the receiver design would favor a simpler single-carrier modulation. For frequency selective channels, the receiver design would prefer multi-carrier modulation as the channel equalization can be performed on each subcarrier.

1.2 Hardware Design Constraints and Tradeoffs

Energy-efficient data representation in application specific baseband transceiver hardware are in demand resulting from energy costs involved in baseband signal processing [1]. In macrocell base stations, about ten percent of energy cost comes from digital signal processing (DSP) modules while power amplification and cooling processes consume more than 70% of total energy [2]. The energy consumption by DSP modules relative to power amplification and cooling will increase in future designs of small cell systems because low-powered cellular radio access nodes handle a shorter radio range [2]. The design of an energy-efficient number representation will reduce overall energy consumption in base stations.

The design choices for the digital radio transceiver subsystems are often made between the tradeoffs of hardware resources and system performance. The hardware resources may include the power, timing and memory requirement, while the system performance could be judged by quantitative measures such as power consumption, clocking frequency, processing bandwidth, supported numeric data types, processing delay, etc.

1.3 Digital Signal Processor Architecture

One of the techniques for keeping the power consumption low is by limiting the clock frequencies. However, limiting the clock frequencies may not be the optimal tradeoff to make against the system performance. The clock frequencies are needed to provide sufficient sampling operations demanded by the software applications. Therefore, the power requirement correlates with the required clock frequencies and sampling frequencies at the analog-to-digital converters (ADC) and digital-to-analog converters (DAC).

Several processor architectures such as SIMD and VLIW parallelism have been proposed to increase overall system performance at the cost of higher power consumption. With VLIW parallelism, several functional units could be initiated to operate on arithmetic operations, logic operations, or memory accesses independently to increase system throughput. With SIMD parallelism, the instruction overheads could be reduced by executing operation on a group of data of similar type. A slightly lower power consumption could be achieved by individually optimizing each of the frequently used functional units.

The floating-point processing unit is included to speed up the processing of scalar/vector floating-point data. The floating-point unit could occupy up to 40% - 50% of the total core area and could consume up to 50% - 60% of the available core power [3].

1.4 Data Converter Architecture

The processor clock rate (cycle per second) is limiting the processor while memory bandwidth (bits per second) is limiting the interaction between processor, memory banks, and arithmetic processing module. Analog-to-digital converters (ADCs) generate discrete and quantized bits to represent the signal magnitude. Oversampling above the Nyquist rate at the data converter is often needed to ensure fidelity of the data signal. The resampling operation will often be needed and performed on the raw data bits when the processor runs on limited resources.

The main design choices of data converter would affect the power consumption, signal bandwidth, and effective number of bits. These parameters are tied together by a general measure of power efficiency of the data converter, namely the figure of merit (FoM). The Walden FoM measures the above quantities and generalizes the energy needed for each conversion step of least significant bit, [4]. The Schreier FoM includes the term dynamic range of the data converter, which is affected by the variable in-band noise power [5]. Extensive consideration of the ADC selection will require some insight on the impact of system design such as placement of anti-aliasing filter, out-of-band signal transmission, and signal leakage [6].

Suppose the passband waveform occupies bandwidth, W, centered at carrier frequency, f_c. The data converter unit would require sampling frequency to be at least, $f_s > W$ since the baseband bandwidth is $B = \frac{W}{2}$ and the Nyquist sampling criterion requires $f_s \geq 2B$.

1.5 Organization

This report is organized in the following format. Chapter 2 reviews the binary floating-point number format most commonly implemented in modern processor. The discussion of complex valued floating-point number format is natural extension that relates to modified floating-point format in prior arts.

Chapter 3 describes the main contribution of this work which focuses on proposed complex block floating-point representation that has acceptable implementation complexity and reduced error in complex block multiplication-based arithmetic. The proposed representation is shown to apply to implementation of fast Fourier transform algorithm in radix-2 and radix-4 format.

Chapter 4 focuses on the system model of communication systems design that features the proposed representation. The particular modulation systems in discussion are single-carrier and multi-carrier amplitude-based modulation which are commonly used in communication technology.

Chapter 5 shows the simulation results obtained from computations of complex block arithmetic and algorithms modeling. Chapter 6 draws on the conclusion and future work based on the design approach and simulation results of the proposed complex block floating-point representation.

Chapter 2

Number Systems

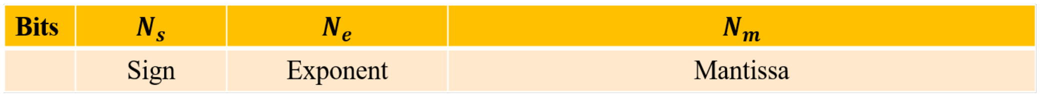
2.1 IEEE-754 Floating-Point Format

2.1.1 Binary Numeric Representation

The IEEE-754 Standards [7] specify a normalized number format that provides better numeric precision and larger range than a fixed-point number systems. Table 2.1 lists a number of supported floating-point number representation in the IEEE-754 Standards. Each supported wordlength is even integer multiple of a byte (8-bit). N_w-bit is defined as the wordlength of scalar floating-point number, N_s as the 1-bit Sign, N_e as the Exponent bit width, and N_m as the Mantissa bit width. The possible value of N_w is the sum of $N_s + N_e + N_m$. Figure 2.1 shows the bit packing of N_w-bit IEEE-754 real-valued number that would be in memory or in floating-point register.

Components	**Definition**	**Bit Widths, B**
Wordlength	N_w	$\{16, 32, 64, 128\}$
Sign	N_s	$\{1, 1, 1, 1\}$
Exponent	N_e	$\{5, 8, 11, 15\}$
Mantissa	N_m	$\{10, 23, 52, 112\}$

Table 2.1: Definition & bit widths of IEEE-754 Format [7] listing half-precision (16-bit), single-precision (32-bit), double precision (64-bit), and quad precision (128-bit). The wordlength is obtained by summing the sign, exponent, and mantissa bits.



Bits	N_s	N_e	N_m
	Sign	Exponent	Mantissa

Figure 2.1: Typical bit packing of IEEE-754 format.

The Mantissa bits are interpreted as the fractional value $(1.0000 \leq x < 2.0000)$ and represent the most significant figures of a number. The IEEE-754 Standards require the normalized result to have a leading bit of 1. The leading bit of 1 is not coded and would be located before the radix-2 point of the Mantissa bits. The Exponent bits are adjusted in the process of normalization and serve as a Base-2 integer/fractional multiplier that controls the magnitude of a number.

To represent numbers that are out of range, the IEEE-754 Standards have specified a few unique sequence. With the Exponent bits set to all ones and the Mantissa bits set to all zeros, the $+\infty$ and $-\infty$ are represented depending on the Sign bit. With the Exponent bits set to all ones and the Mantissa bits set to any non-zeros, Not a Number (NaN) is represented. To represent an exact 0.0, the Exponent bits are set to all zeros and the Mantissa bits are set to all zeros. The Sign bit of 0 determines +0.0 and the Sign bit of 1 determines -0.0.

There are several quantitative measures that determine the effectiveness and show the limitation of this number format. The numeric range is defined as the most negative and the most positive values that can be represented. The numeric precision is defined as the smallest mantissa increment that affects the resultant mantissa value in addition/subtraction with a value of 1.0. The effective dynamic range, D, is calculated as the ratio of numeric range to numeric precision, $D = 20log(\frac{numeric\ range}{precision})$. With a specific signal-to-noise ratio requirement in an application, the effective dynamic range would be a useful metric to compare and select from the IEEE-754 format and non-IEEE-754 format.

2.1.2 Implementation Complexity

The IEEE-754 Standards also specify the exception handling requirement of invalid operation, division by zero, overflow and underflow. Algorithms that consist of heavy multiplication and addition operations of two large numbers or two small numbers are likely to incur overflow or underflow exceptions. To conform with the standards, additional circuitry is needed to perform exception handling.

The two common arithmetic operations involved in signal processing are addition and multiplication. All arithmetic operations that operate on two floating-point numbers require internal rescaling of mantissa and exponent bits.

For the addition operation, the Exponent bits must be compared and the leading hidden bit of one before the Mantissa bit is recovered. If the Exponent bits are not equal, then the smaller Exponent value would be increased by one and the Mantissa value would be divided by two. This process repeats until both Exponent values are equal. Fixed-point addition is applied on the Mantissa bits to give an intermediate Mantissa. There will be tradeoff made between the delay and gate cost involved in the fixed-point addition algorithm. For example, with a carry skip adder algorithm, the delay of the Mantissa bits addition result can be reduced to $O(\sqrt{N_m + 1})$ at the gate cost of $O(N_m)$. The intermediate Mantissa bits would be normalized to give a leading bit of one, increasing/decreasing the value of Exponent bits in the process.

For the multiplication operation, the pre-processing of Exponent and Mantissa bits is less complex. The leading 1s of Mantissa bits are first recovered. Fixed-point multiplication is applied to the Mantissa bits and fixed-point addition is applied to the Exponent bits. For example, with the Dadda reduction technique, the complexity of the Mantissa bits multiplication operation can be reduced to approximately

$O(log_{1.45}(N_m+1))$ stages to use $O(2(N_m))$ carry lookahead adder, $O(N_m^2)$ full adders, and $O(N^{1.5})$ half adders. The delay would account for $O(log_{1.45}(N_m))$ stages of full adder delays and $O(log_r(N_m))$ carry lookahead adder delay. For example, with carry skip adder algorithm, the delay of the Exponent bits addition result can be reduced to $O(\sqrt{N_e})$ at the gate cost of $O(N_e)$. The intermediate Mantissa bits would also undergo normalization process to give leading bit of one and affect the value of the Exponent.

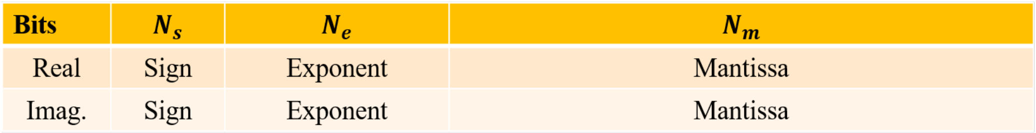
Both floating-point addition and multiplication require normalization of the intermediate Mantissa and Exponent to produce the final result. The floating-point addition requires two pre-scaling of input Mantissa and Exponent bits, but only involves one fixed-point addition of the Mantissas. Floating-point multiplication does not require pre-scaling of input Mantissa or Exponent bits, but this involves two operations: one fixed-point addition of Exponents and one fixed-point multiplication of Mantissas.

The selection of addition and multiplication algorithms will depend on the actual bit width of N_e for arithmetic on the Exponents and N_m for arithmetic on the Mantissas. That is because certain algorithms may provide speedup for particular bit widths, i.e., root of a number, even numbers, or odd numbers. With the IEEE-754 Standard, the half-precision format has an odd number of Exponent bits and (Mantissa + 1) bits, the single-precision format has an even number of Exponent bits and (Mantissa + 1) bits, the double-precision format has an odd number of Exponent bits and (Mantissa + 1) bits, and the quad-precision format has an odd number of Exponent bits and (Mantissa + 1) bits. If the selection of addition and multiplication algorithms are not optimized based on the number of input bits, then the input bits are not fully utilized in the arithmetic operations.

In any arithmetic operations that potentially change the output values of Mantissas, the Mantissas of the result will undergo a stage of re-normalization and the Exponent will increment or decrement correspondingly.

2.1.3 Complex-valued Representation

Each complex number can be thought of as two orthogonal real numbers in a Cartesian coordinate system. The arithmetic operations are defined in complex number domain. Each complex number addition can be realized by performing two real number additions. Each complex number multiplication can be realized by performing four real number multiplications and two real number additions. With simple extension of IEEE-754 format from one real value dimension to two real values dimension, a complex floating-point number can be simply coded as a pair of floating-point numbers with similar bit widths and precision. This would allow hardware reuse or easy replication of hardware units. Figure 2.2 shows the bit packing of a complex IEEE-754 number that would require $2N_w$-bits in memory or in floating-point register where N_w is the scalar floating-point bit width.

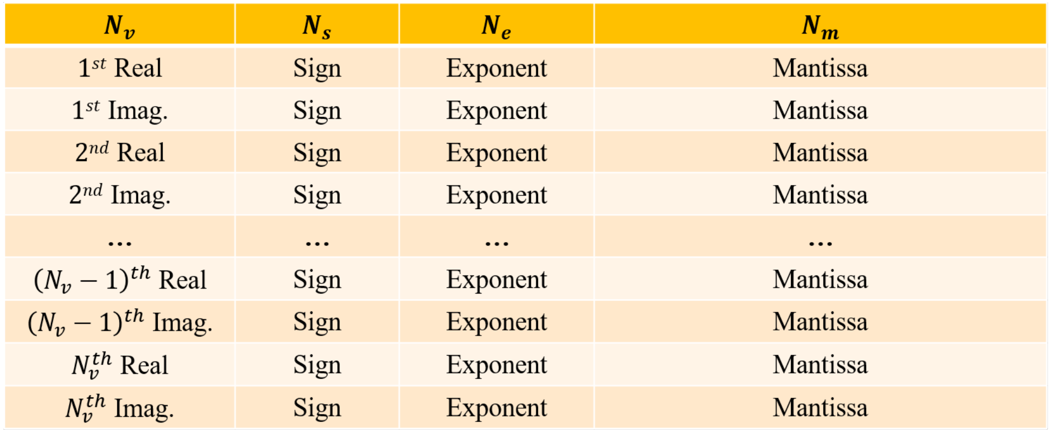


Bits	N_s	N_e	N_m
Real	Sign	Exponent	Mantissa
Imag.	Sign	Exponent	Mantissa

Figure 2.2: Typical bit packing of complex floating-point in IEEE-754 precision. Each real/imaginary dimension has the IEEE-754 precision.

2.2 Complex Block Floating-Point Format

While information is rarely contained in just a scalar real/complex number, the IEEE-754 numeric representation is often used to represent an array of real/complex numbers. Suppose N_v is the number of complex samples in an array, Figure 2.3 shows the bit packing of an array of complex IEEE-754 numbers that would require $2N_vN_w$-bits in memory or in vector floating-point register.



N_v	N_s	N_e	N_m
1^{st} Real	Sign	Exponent	Mantissa
1^{st} Imag.	Sign	Exponent	Mantissa
2^{nd} Real	Sign	Exponent	Mantissa
2^{nd} Imag.	Sign	Exponent	Mantissa
…	…	…	…
$(N_v - 1)^{th}$ Real	Sign	Exponent	Mantissa
$(N_v - 1)^{th}$ Imag.	Sign	Exponent	Mantissa
N_v^{th} Real	Sign	Exponent	Mantissa
N_v^{th} Imag.	Sign	Exponent	Mantissa

Figure 2.3: Typical vector form bit packing of IEEE-754 complex block. Each complex sample has twice the IEEE-754 precision.

The complex block floating-point format is preferred for the purpose of improving both the throughput and latency of obtaining the block arithmetic results and reducing the area and implementation complexity of block arithmetic units. Suppose the information encoded in a complex block has continuity in magnitude/phase, a complex block floating-point format can be designed that reduces the wordlength requirements. The idea of coherence in magnitude or phase is shown in Figure 2.4.

The discrete samples within in a time window may locate within a region or boundary on a complex value plane. A practical example that shows coherence in magnitude and phase is the randomly generated 64-QAM symbols oversampled by 4x above the Nyquist rate in Figure 2.5. With oversampling condition, this shows that any time-delayed samples generated by the same data converter source would have the characteristic of coherence in magnitude and phase.

The tradeoff associated with reducing the wordlength maybe the reduced mag-

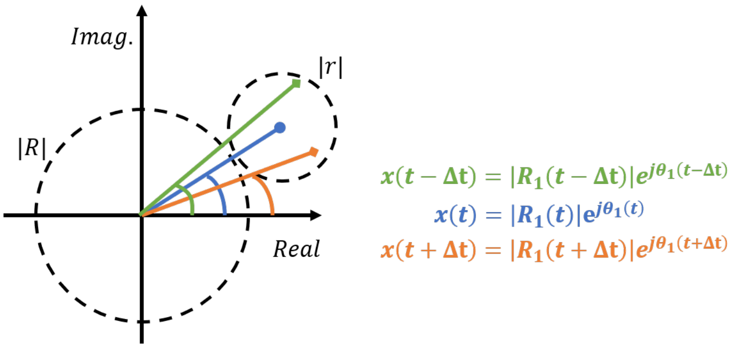

Imag.
$|r|$
$|R|$
Real
$x(t-\Delta t) = |R_1(t-\Delta t)|e^{j\theta_1(t-\Delta t)}$
$x(t) = |R_1(t)|e^{j\theta_1(t)}$
$x(t+\Delta t) = |R_1(t+\Delta t)|e^{j\theta_1(t+\Delta t)}$

Figure 2.4: Multiple discrete-time complex samples on the scatter plot of real/imaginary dimension. The smaller dotted circle enclosed complex samples that may have coherence in both magnitude and phase due to high sampling rate.

nitude/phase resolution in the polar representation. A more memory efficient complex block format could be designed such that the bits allocation are used to encode the magnitude or phase difference in a complex block floating-point number instead of the absolute magnitude/phase value of each complex number in the block.

In hardware design, the data converter converts continuous-time input signals and generates digital amplitude representations at uniformly spaced time intervals. Data converter design on this complex block floating-point format will be described in detail.

2.2.1 Common Exponent Encoding

The concept of common exponent encoding is shown in scalar complex floating-point representation in [8] and block floating-point representation in [9]. The authors in [8] target an improvement on IEEE-754 16-bit half-precision and show a complex floating-point representation that requires 29-bits and achieves 3-bit effective savings. The 3-bit savings is achieved by saving one of the 5-bit Exponent (N_e)-bit while reusing 2-bits for explicitly encoding the leading bit of "1" or "0" for re-normalized

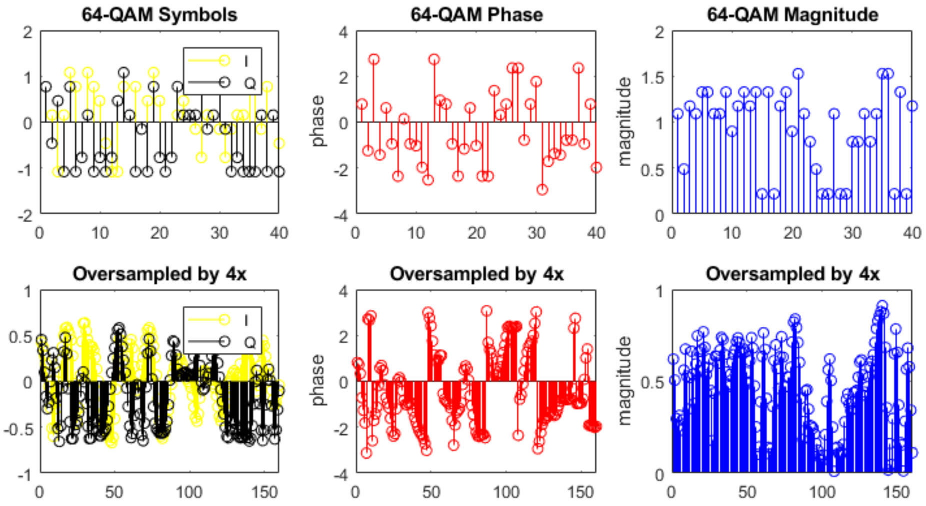


Figure 2.5: [Top row, from the left to right] The in-phase/quadrature-phase (IQ) waveform consists of 64-QAM symbols in 40 symbols length. The phase and magnitude plots of the IQ waveform of 64-QAM symbols show little phase coherence and more magnitude coherence. [Bottom row, from the left to right] The 4x oversampled waveform of the 64-QAM symbols in 160 samples length. The phase and magnitude plots of the upsampled waveform show the most phase and magnitude coherence.

numbers. The Mantissas are 11-bit (N_m+1)-bit and all coded. The authors also report ASIC silicon footprint of arithmetic units (adder, multiplier) are larger by 10% and registers and memories footprint are smaller by 10 % in its VLSI implementation. The performance loss reported varies between 0.2 dB in low SNR and 2 dB in high SNR cases.

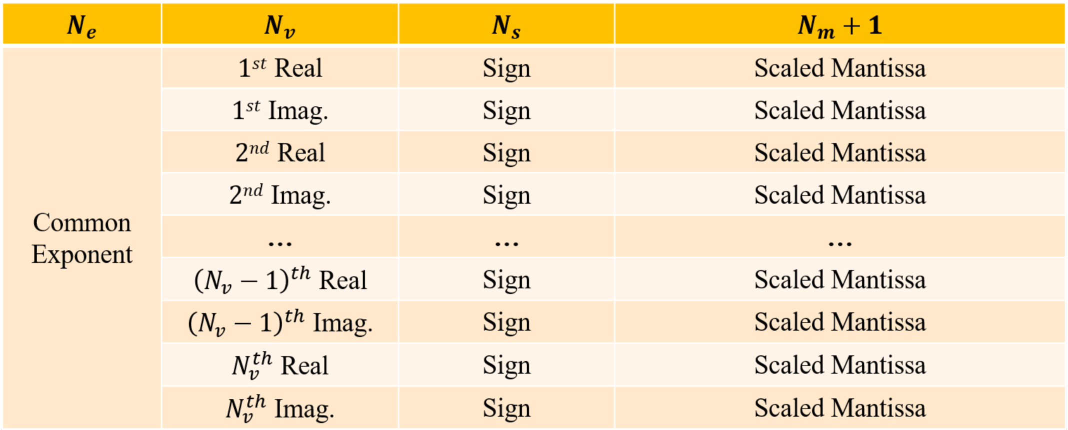
The authors in [9] emulate the block floating-point algorithm on a fixed-point digital signal processor for a fast Fourier Transform arithmetic unit. The authors implement a 64-point FFT through radix-4 1st stage and radix-2 subsequent stages in the block floating-point format. The authors report an error measurement better than fixed-point FFT, which is expected. However, the more helpful piece of error measurement should be documented relative to purely floating-point FFT output accuracy. There is no mention of floating-point precision nor IEEE-754 relevance.

Table 2.2 shows the wordlength required for N_v complex samples per block in common exponent encoding method. Figure 2.6 shows the bit packing of N_v complex samples in an array that is in memory or in vector floating-point register. The behavior of the common exponent encoding in a complex block format would have a somewhat similar behavior as polar representation of complex number encoding, although not exactly, in terms of bit allocation. The most significant sample in terms of magnitude, either the real part or the imaginary part would get the full bit representation, $(N_m + 1 + 2^{N_e})$-bits in fixed-point equivalent precision, without any quantization loss relative to the IEEE-754 format. All other samples would only get partial bit representation, $(N_m + 1 - 2^{\Delta N_e})$-bits in fixed-point equivalent precision. The smaller complex values in the block reduces the mantissa precision since the process of re-normalizing to a common exponent in the block introduces more leading zero bits.

Components	**Definition**	**Bit Widths, B**
Wordlength	N_w	$\{5, 8, 11, 15\} + (2 * N_v) * \{12, 25, 54, 114\}$
Common Exponent	N_e	$\{5, 8, 11, 15\}$
Sign	N_s	$\{1, 1, 1, 1\}$
Mantissa	N_m	$\{11, 24, 53, 113\}$

Table 2.2: Definition & bit widths of common exponent encoding assuming block size of N_v complex samples per block. The wordlength is obtained by summing a common exponent, $2N_v$ times sign and mantissa bits. The mantissa bit is one bit wider than IEEE-754 format to ensure the dominant value has same IEEE-754 precision.

Figure 2.7 describes the algorithm used in conversion of complex block format, assuming that IEEE-754 format is first available. Essentially, the common exponent encoding technique applied to complex block floating-point format allows sharing of the common exponent in a block of complex-valued data. The mantissa pairs are re-normalized to the value of the common exponent after the leading bit is recovered.



N_e	N_v	N_s	$N_m + 1$
Common Exponent	1^{st} Real	Sign	Scaled Mantissa
	1^{st} Imag.	Sign	Scaled Mantissa
	2^{nd} Real	Sign	Scaled Mantissa
	2^{nd} Imag.	Sign	Scaled Mantissa
	…	…	…
	$(N_v - 1)^{th}$ Real	Sign	Scaled Mantissa
	$(N_v - 1)^{th}$ Imag.	Sign	Scaled Mantissa
	N_v^{th} Real	Sign	Scaled Mantissa
	N_v^{th} Imag.	Sign	Scaled Mantissa

Figure 2.6: Typical vector form bit packing of common exponent format complex block. The common exponent applies to all complex samples in the block.

Each of the complex pair would trade the exponent bits with 1-bit leading bit for more memory storage space. Although being memory efficient, the digital representation would have weaker encoding of phase resolution in each of the complex pair in the block. The maximum allowed exponent difference without huge quantization error under the common exponent encoding is derived in Appendix A. The mantissas of either the real part or the imaginary part could be reduced to zeros as a result of large phase difference in a complex sample. It is also possible that the mantissas of smaller value complex samples in a block be reduced to all zeros as a result of large magnitude difference across multiple complex samples.

The amount of bit savings in terms of percentage depends on the number of bits per floating-point number and the block size. Common exponent encoding would achieve the highest amount of savings for half-precision floating-point number and smallest amount of savings for quad-precision floating-point number. This is due to the averaging effect by the exponentially growing word size in the IEEE-754 table. Table 2.3 lists the expected bit savings for varying IEEE-754 precision numbers with 64 complex samples per block.

<u>Conversion to Common Exponent Encoding</u>

Step 1: $$E_{common} = \max\{Real(\vec{E}), Imag(\vec{E})\}$$

Step 2: $$\Delta\, Real(\vec{E}) = E_{common} - Real(\vec{E})$$ $$\Delta\, Imag(\vec{E}) = E_{common} - Imag(\vec{E})$$

Step 3: $$Real(\vec{M}) = Real(1.\vec{M}) \gg \Delta\, Real(\vec{E})$$ $$Imag(\vec{M}) = Imag(1.\vec{M}) \gg \Delta\, Imag(\vec{E})$$

Step 4: Done

Figure 2.7: The algorithm assumes IEEE-754 format is first available and consists of four main steps. Step 1 identifies the common exponent. Step 2 identifies the amount of right shifting on the mantissa which is the difference between common exponent and original real/imaginary exponents. Step 3 recovers the leading 1 bit of real/imaginary mantissa and performs appropriate right shifting. Step 4 completes the conversion.

IEEE-754	# of Exponent Saved	# of Traded	# Overall Saved Per Block / %
16-bit	5-bit	2-bit	(3+8*63)-bit / 24.76%
32-bit	8-bit	2-bit	(6+14*63)-bit / 21.68%
64-bit	11-bit	2-bit	(9+20*63)-bit / 15.49%
128-bit	15-bit	2-bit	(13+28*63)-bit / 10.85%

Table 2.3: The expected bit savings in memory and register for common exponent format. With block size of 64 complex samples per block, the bit savings range between 10.85% and 24.76% depending on the chosen IEEE-754 precision.

The effective encoding region is defined as a square of size $(N_m \times N_m)$ with the chosen common exponent at the top right corner. The out of touch region floats next to the effective encoding region and bounded by the real-axis and imaginary axis. The out of touch region contains all complex exponent pairs in which the corresponding IEEE-754 mantissa are potentially zeroed as a result of aggressive rescaling. The union of the effective encoding region and the out of touch region forms a subset of the entire encoding space $(2^{N_m} \times 2^{N_m})$ of the IEEE-754 format as shown in Figure 2.8.

The effective encoding region is labeled (□) in Figure 2.8 and encloses only

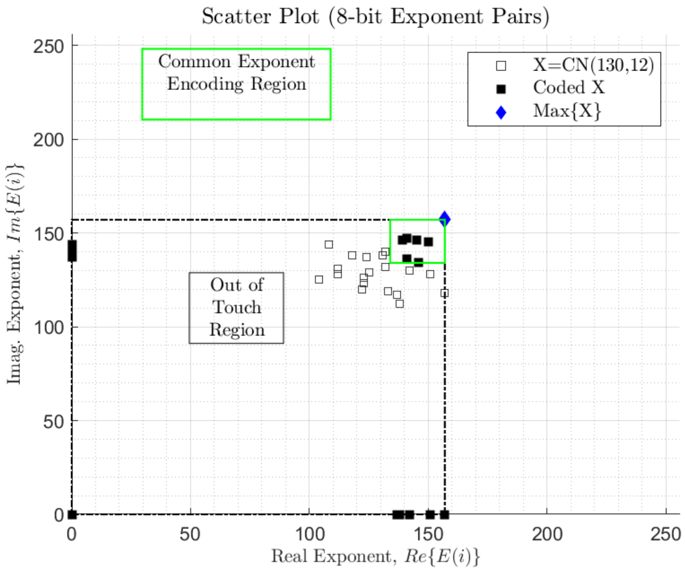


Figure 2.8: The unfilled rectangle symbols indicate the original real/imaginary exponent pairs. The dark rectangle symbols indicate coded exponent pairs. The largest exponent value is selected to be the common exponent at the top right corner of the effective encoding region. The size of effective encoding region is the square of the mantissa bit width, $N_m + 1$. Any complex exponent pairs that fall outside the effective encoding region will have the corresponding mantissa values coded as zeroes. The possible locations of coded exponent pairs are within the effective encoding region, on the x-axis, y-axis, and at the origin.

a fraction of the 25 normally distributed complex valued exponent pairs with mean of 130 and variance of 12. The empty rectangular white box symbols indicate complex exponent pairs in separate exponents encoding. The solid rectangular filled box symbols indicate complex exponent pairs in common exponent encoding. The coded complex exponent pairs that fall on either y-axis, x-axis, or origin imply that one or both of real/imaginary mantissa values are coded as zeros.

The accuracy of the complex block arithmetic (adder, multiplier, matrix operations) would rely on each mantissa values in the complex block format. The common exponent encoding is expected to have higher quantization error in the complex block format and block arithmetic output since the performance of the complex block representation is data dependent.

2.2.2 Conversion with Two Parallel ADCs

Figure 2.9 describes a block diagram with two parallel data converters that are assumed to sample in time and provide digital representation of two stream of numbers in fixed-point format, namely the real and imaginary dimension of complex numbers. The automatic gain control applies gain on the analog inputs prior to the data converter with the feedback information on the quantized output of the data converter. The data converter output passes fixed-point number to convert into higher precision floating-point number. The parallel stream of real and imaginary floating-point numbers are combined to generate complex floating-point number in a block. To benefit from memory efficient storage such as common exponent encoding, a 2nd stage of number conversion is needed.

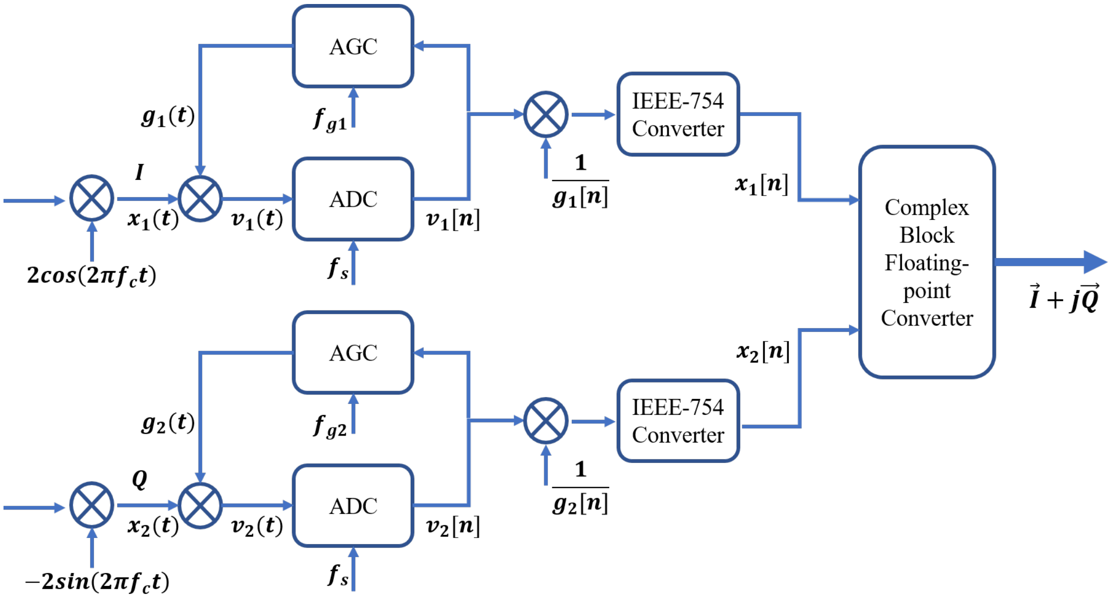


Figure 2.9: Two analog-to-digital converters (ADCs) running in parallel generate the in-phase/quadrature-phase waveforms. To convert to floating-point format, both ADCs work with separate automatic gain control (AGC) to maximize signal-to-quantization-noise ratio (SQNR). In this architecture, there is a need of second stage conversion to complex block floating-point format which applies the algorithm in Figure 2.7.

2.2.3 Conversion with Single ADC

Figure 2.10 describes a different front end processing chain that has one data converter. The analog inputs to the ADC have passband signals centered at intermediate frequency, $f_{Im} > 0$. This specific design eliminates the time-synchronization issue between two parallel data converter chains that would occur in Figure 2.9 in practice.

The complex block floating-point converter is the only number conversion stage needed in the architecture. The two-stage number conversion in the Figure 2.9 have been transformed to two-stage downconversion in this architecture. The first downconversion is performed in continuous-time and the second downconversion is performed in discrete-time fixed-point format. The inputs to the converter undergo fixed-point multiplication prior to the actual conversion. It is possible to use the gain values produced by the automatic gain control block as the common exponent, therefore, the complex floating-point converter and the automatic gain control block could share a common clock for synchronization purpose.

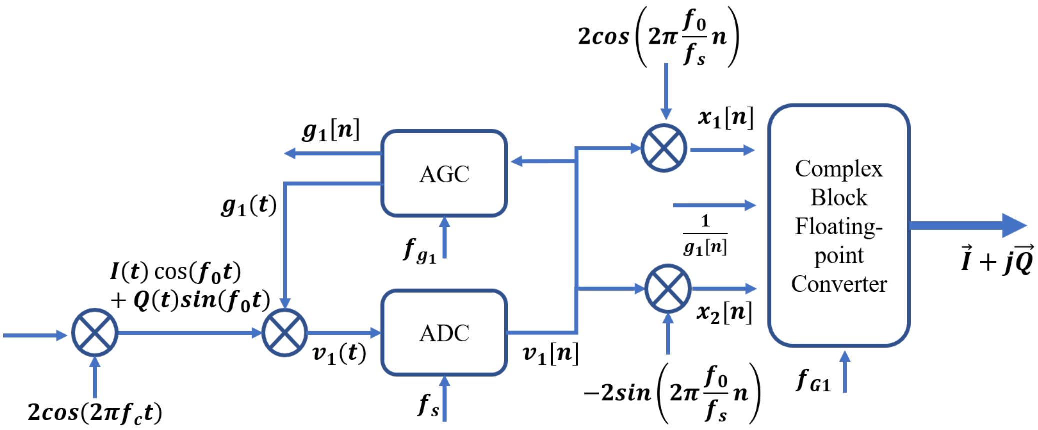


Figure 2.10: One ADC converts complex in-phase/quadrature-phase (complex IQ) waveform to complex block format directly. The common exponent applied to the complex block is the inverse of gain value of the automatic gain controller (AGC). The IQ waveform needs a separate demodulation stage in discrete-time prior to conversion to complex block format.

Chapter 3

Proposed Complex Block Floating-Point Representation

3.1 Exponent Box Encoding Technique

The main issue with common exponent complex block floating-point format is the loss of amplitude and phase precision due to aggressive rescaling of mantissas. A new complex block representation is proposed to address this issue in block floating-point representation. Table 3.1 shows the wordlength required for N_v complex samples per block with exponent box encoding extension.

Components	**Definition**	**Bit Widths,** B
Wordlength	N_w	$\{5, 8, 11, 15\} + (2 * N_v) * \{13, 26, 55, 115\}$
Common Exponent	N_e	$\{5, 8, 11, 15\}$
Sign	N_s	$\{1, 1, 1, 1\}$
Box Shift	N_x	$\{1, 1, 1, 1\}$
Mantissa	N_m	$\{11, 24, 53, 113\}$

Table 3.1: Definition & bit widths of exponent box encoding assuming block size of N_v complex samples per block. The wordlength is obtained by summing a common exponent, $2N_v$ times sign, box shift, and mantissa bits.

It is noted that two additional bits per complex sample in a block are needed in the new format relative to common exponent encoding. From the exponent bits saved, the reuse of 2-bit per complex-valued pair is shown to reduce quantization error and improve accuracy of block arithmetic result [10]. The block arithmetic complexity does not significantly increase in new complex block representation [10]. Figure 3.1 shows the bit packing of N_v complex samples in an array that is in memory or in vector floating-point register.

Figure 3.2 describes the algorithm used in conversion of complex block format,

N_e	N_v	N_s	N_x	$N_m + 1$
Common Exponent	1^{st} Real	Sign	Box Shift	Scaled Mantissa
	1^{st} Imag.	Sign	Box Shift	Scaled Mantissa
	2^{nd} Real	Sign	Box Shift	Scaled Mantissa
	2^{nd} Imag.	Sign	Box Shift	Scaled Mantissa
	…	…	…	…
	$(N_v - 1)^{th}$ Real	Sign	Box Shift	Scaled Mantissa
	$(N_v - 1)^{th}$ Imag.	Sign	Box Shift	Scaled Mantissa
	N_v^{th} Real	Sign	Box Shift	Scaled Mantissa
	N_v^{th} Imag.	Sign	Box Shift	Scaled Mantissa

Figure 3.1: Typical vector form bit packing of exponent box format complex block. The exponent box shift is 2-bit per complex sample.

assuming that IEEE-754 format is first available and this conversion applies to ADC architecture described in Figure 2.9, Section 2.2.2. This recommendation is preferred in implementation since the exponent is at least 5-bit for half-precision and up to 15-bit for quad-precision in IEEE-754 Standard. For all listed IEEE-754 precision, the expected bit savings per block is tabulated in Table 3.2.

Conversion to Exponent Box Encoding

Step 1: $E_{common} = \max\{Real(\vec{E}), Imag(\vec{E})\}$

Step 2 (a):

$$\Delta\, Real(\vec{E}) = E_{common} - Real(\vec{E})$$
$$\Delta\, Imag(\vec{E}) = E_{common} - Imag(\vec{E})$$
$$Real(\vec{X}) = \Delta\, Real(\vec{E}) > B_m$$
$$Imag(\vec{X}) = \Delta\, Imag(\vec{E}) > B_m$$

Step 2 (b):

If $(Real(\vec{X}))$: $\Delta\, Real(\vec{E}) = \Delta\, Real(\vec{E}) - B_m$
If $(Imag(\vec{X}))$: $\Delta\, Imag(\vec{E}) = \Delta\, Imag(\vec{E}) - B_m$

Step 3:

$$Real(\vec{M}) = Real(1.\vec{M}) \gg \Delta\, Real(\vec{E})$$
$$Imag(\vec{M}) = Imag(1.\vec{M}) \gg \Delta\, Imag(\vec{E})$$

Step 4: Done

Figure 3.2: The algorithm is similar as the conversion to common exponent format in Figure 2.7. Step 2(a) marks box shift bit to be 1 if the right-shifting of mantissa will remove all the bits. Step 2(b) reduces the amount of right-shifting by mantissa bit width, B_m.

Figure 3.3 compares the effective encoding region in exponent box encoding technique (□) to the common exponent encoding technique (□). The solid rectangu-

IEEE-754	# of Exponent Saved	# of Traded	# Overall Saved Per Block / %
16-bit	5-bit	4-bit	(1+6*63)-bit / 18.51%
32-bit	8-bit	4-bit	(4+12*63)-bit / 18.55%
64-bit	11-bit	4-bit	(7+18*63)-bit / 13.93%
128-bit	15-bit	4-bit	(11+26*63)-bit / 10.06%

Table 3.2: The expected bit savings in memory and register for exponent box format. With block size of 64 complex samples per block, the bit savings range between 10.06% and 18.51% depending on the chosen IEEE-754 precision.

lar filled box symbols indicate effectively coded complex exponent pairs in the new format. The empty rectangular white box symbols indicate the exponent pairs such that mantissa values become zeros in the new format. In the new format, majority of the complex exponent pairs are effectively coded.

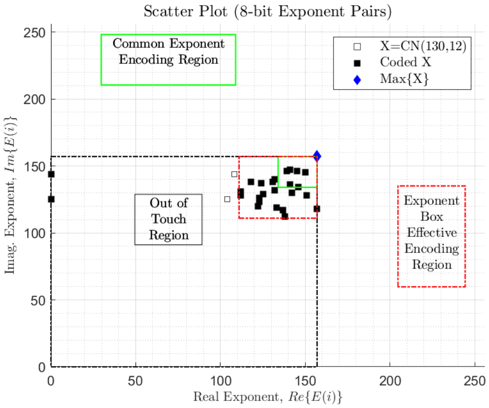


Figure 3.3: The plot is similar as effective encoding region for common exponent format in Figure 2.8. The effective encoding region is 4x larger than that of common exponent format. Any coded exponent pairs can fall into one of the four regions. The amount of coded exponent pairs on x-axis, y-axis, and at the origin reduces.

In common exponent encoding, the size of the effective encoding region is fixed by the mantissas bit width. In the exponent box encoding, the 2-bit shift values per complex pair allows the exponent box float vertically, horizontally, or diagonally from the common exponent. The effective encoding region becomes four times larger than that of the common exponent encoding.

3.2 Vector Arithmetic Subsystem Modeling

Vector processing unit is common for performing arithmetic operation that increases the subsystems throughput with parallelism of multiple scalar arithmetic subsystems. This subsystem design would require buffering mechanism prior to issue of vector arithmetic operation. An example of vector arithmetic unit is shown in Figure 3.4.

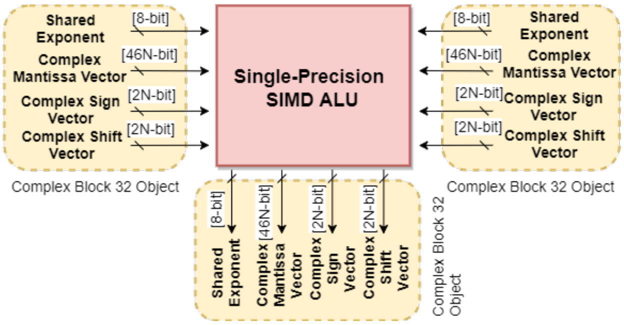


Figure 3.4: The interface of single-precision complex block arithmetic subsystems is shown for the 2 complex input block and 1 complex output block in the exponent box format.

3.2.1 Complex Block Addition

The Equation 3.1 describes the mathematical relation of complex block addition. Let $\boldsymbol{X_1}, \boldsymbol{X_2}, \boldsymbol{Y} \in \mathbb{C}^{1\times N_v}$ be complex-valued vectors with N_v samples per vector, such that,

$$\begin{aligned}\Re\{\boldsymbol{Y}\} &= \Re\{\boldsymbol{X_1}\} + \Re\{\boldsymbol{X_2}\} \\ \Im\{\boldsymbol{Y}\} &= \Im\{\boldsymbol{X_1}\} + \Im\{\boldsymbol{X_2}\}\end{aligned} \tag{3.1}$$

Figure 3.5 shows the simplified block diagram for complex block addition operation. With exponent box encoding, the mantissa vectors have extended precision.

The $2N_v$-bit binary box shift vector per complex block would decide if additional pre-processing is required on the input mantissa vectors. It is also possible to combine the pre-processing stage due to box shift vector and difference in common exponent and reduces the overall pre-processing complexity to $O(N_v)$. Before truncating the output mantissa vectors to bit width of the mantissa precision, the output mantissa vectors are examined whether the box shift vector would be used on the output complex block. The overall post-processing complexity for output mantissa vectors stay at $O(N_v)$.

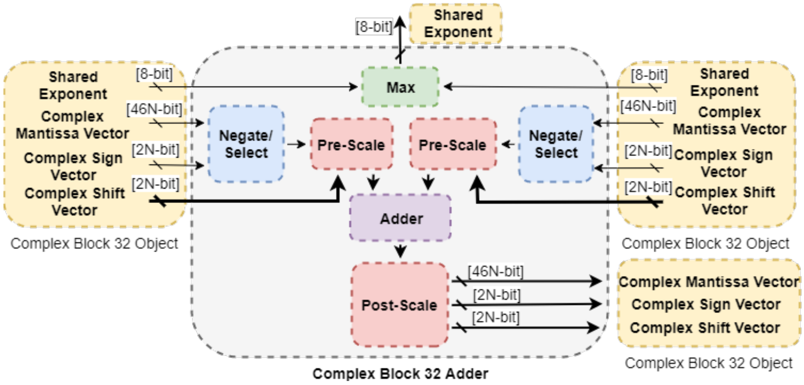


Figure 3.5: The interface of single-precision complex block addition subsystems is shown for the 2 complex input block and 1 complex output block in the exponent box format. The pre-processing steps involve rescaling of mantissa bits based on the exponent box shift value.

Table 3.3 shows the effect of input complex binary shift vectors on the complex block addition. The shorthand notation used here eliminates the index term, k^{th}, i.e. $N_{x,real,1} = N_{x,real,1}(k)$ where $k = \{0,\ ...,\ N_v - 1\}$. The pre-processing of addition operation will compare the two values of input shared exponents and perform right shift on the mantissa vector by mantissa bit width. This operation essentially reverses the quantization into the exponent box encoding. The real valued binary shift vectors of both inputs are used to determine whether right shift on real valued mantissa vectors. The imaginary valued binary shift vectors have the same impact on the imaginary valued mantissa vectors. In all cases, the output exponent is the maximum value of the shared exponent of two input complex block.

Vec1 / Vec2	$N_{x,real,2} == 0$	$N_{x,real,2} == 1$
$N_{x,real,1} == 0$	Do nothing	$N_{m,real,2} >> B_m$
$N_{x,real,1} == 1$	$N_{m,real,1} >> B_m$	$N_{m,real,1}, N_{m,real,2} >> B_m$

Vec1 / Vec2	$N_{x,imag.,2} == 0$	$N_{x,imag.,2} == 1$
$N_{x,imag.,1} == 0$	Do nothing	$N_{m,imag.,2} >> B_m$
$N_{x,imag.,1} == 1$	$N_{m,imag.,1} >> B_m$	$N_{m,real,1}, N_{m,real,2} >> B_m$

Table 3.3: Prior to complex mantissa addition, the exponent box shift bits indicate whether further right-shifting of complex mantissa is needed. The amount of right-shifting is always constant, the mantissa bit width, B_m.

Figure C.1 and Figure C.2 in the Appendix C shows the block diagram of addition unit for exponent box encoding.

3.2.2 Complex Block Multiplication

The Equation 3.2 describes the mathematical relation of complex block multiplication. Let $\boldsymbol{X_1, X_2, Y} \in \mathbb{C}^{1\times N_v}$ be complex-valued vectors with N_v samples per vector, where $\bullet$ denotes element-wise multiply, such that,

$$\begin{aligned}\Re\{\boldsymbol{Y}\} &= \Re\{\boldsymbol{X_1}\} \bullet \Re\{\boldsymbol{X_2}\} - \Im\{\boldsymbol{X_1}\} \bullet \Im\{\boldsymbol{X_2}\} \\ \Im\{\boldsymbol{Y}\} &= \Re\{\boldsymbol{X_1}\} \bullet \Im\{\boldsymbol{X_2}\} + \Im\{\boldsymbol{X_1}\} \bullet \Re\{\boldsymbol{X_2}\}\end{aligned} \tag{3.2}$$

Figure 3.6 shows simplified block diagram for complex block multiplication. In common exponent encoding, the two input common exponents are added to form the intermediate exponent, sum of $N_{e,1} + N_{e,2}$. The common exponent addition is real-valued arithmetic. The intermediate exponent applies to all four pairs of mantissa bits multiplication result, i.e. $\{(N_{m,real,1}+1)(N_{m,real,2}+1),\ (N_{m,imag.,1}+1)(N_{m,imag.,2}+$

$1),\ (N_{m,real,1}+1)(N_{m,imag.,2}+1),\ (N_{m,imag.,1}+1)(N_{m,real,2}+1)\}$.

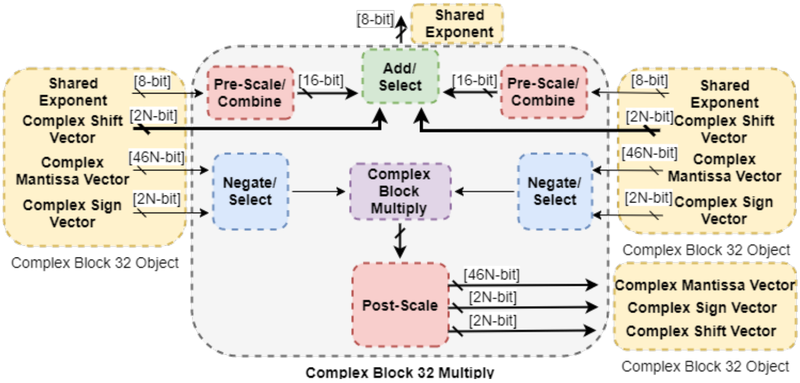


Figure 3.6: The interface of single-precision complex block multiplication subsystems is shown for the 2 complex input block and 1 complex output block in the exponent box format. The pre-processing steps involve computing four different values of intermediate exponents.

With exponent box encoding, the 2-bit complex binary shift values create three possible values of intermediate exponent, $\{N_{e,1}+N_{e,2},\ N_{e,1}+N_{e,2}-B_m,\ N_{e,1}+N_{e,2}-2B_m\}$ where B_m is mantissa bit width. The intermediate mantissa have $2N_m$-precision if there are more than 1 intermediate exponent allowed. Table 3.4 shows the effect of input complex binary shift vectors on the complex block multiplication.

Suppose it is defined that, $A_1 = N_{x,real,1}+N_{x,real,2}$, $B_1 = N_{x,imag.,1}+N_{x,imag.,2}$, $C_1 = (N_{m,real,1}+1)(N_{m,real,2}+1)$, $D_1 = (N_{m,imag.,1}+1)(N_{m,imag.,2}+1)$, and $N_{e,int,real} = N_{e,1}+N_{e,2}-K_1B_m$, then Table 3.4 lists the possible values of complex binary shift vectors effecting the real intermediate exponents and mantissas prior to addition is given as follow, where X is a don't care term and U is an unknown term:

The similar operation can be applied to imaginary intermediate exponent, mantissa generation in the complex block multiplication. Suppose it is defined that, $A_2 = N_{x,real,1}+N_{x,imag.,2}$, $B_2 = N_{x,imag.,1}+N_{x,real,2}$, $C_2 = (N_{m,real,1}+1)(N_{m,imag.,2}+1)$, $D_2 = (N_{m,imag.,1}+1)(N_{m,real,2}+1)$, and $N_{e,int,real} = N_{e,1}+N_{e,2}-K_2B_m$, then Table 3.5 also lists the possible values of complex binary shift vectors effecting the imaginary

A_1	B_1	$K_1 = min(A_1, B_1)$	C_1	D_1
0	X	0	C_1	$D_1 >> (X - K_1)B_m$
1	0	0	$C_1 >> 1B_m$	D_1
1	U	min(1,U)	$C_1 >> (1 - K_1)B_m$	$D_1 >> (U - K_1)B_m$
2	0	0	$C_1 >> 2B_m$	D_1
2	1	1	$C_1 >> (2 - 1)B_m$	D_1
2	U	min(2,U)	$C_1 >> (2 - K_1)B_m$	$D_1 >> (U - K_1)B_m$

Table 3.4: The complex mantissa multiplication generates intermediate mantissa values. Prior to mantissa addition, each intermediate mantissa is scaled to the intermediate exponent value to preserve more bits. The intermediate exponent values are selected based on the input shared exponents and individual exponent box shift bits.

intermediate exponents and mantissas prior to addition is given as follow:

A_2	B_2	$K_2 = min(A_2, B_2)$	C_2	D_2
0	X	0	C_2	$D_2 >> (X - K_2)B_m$
1	0	0	$C_2 >> 1B_m$	D_2
1	U	min(1,U)	$C_2 >> (1 - K_2)B_m$	$D_2 >> (U - K_2)B_m$
2	0	0	$C_2 >> 2B_m$	D_2
2	1	1	$C_2 >> (2 - 1)B_m$	D_2
2	U	min(2,U)	$C_2 >> (2 - K_2)B_m$	$D_2 >> (U - K_2)B_m$

Table 3.5: The complex mantissa multiplication generates intermediate mantissa values. Prior to mantissa addition, each intermediate mantissa is scaled to the intermediate exponent value to preserve more bits. The intermediate exponent values are selected based on the input shared exponents and individual exponent box shift bits.

Figure C.3 and Figure C.4 in the Appendix C shows the block diagram of multiplication unit for exponent box encoding.

3.3 Algorithms Modeling

3.3.1 Fast Fourier Transform (Radix-2)

Suppose that N is a base-2 number. The Fourier transform formula can be written in terms of two $\frac{N}{2}$-point FFT in Equation 3.3 with the following twiddle factors, $W_N = e^{-j\frac{2\pi}{N}}$ and radix number, $r = 2$.

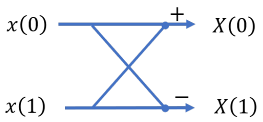
$$\begin{aligned} S_k &= \sum_{m=0}^{N-1} s_m W_N^{mk} \\ &= \sum_{m=0}^{\frac{N}{2}-1} s_m W_N^{mk} + \sum_{n=\frac{N}{2}}^{N-1} s_n W_N^{nk} \\ &= \sum_{m=0}^{\frac{N}{2}-1} s_m W_N^{mk} + \sum_{n=0}^{\frac{N}{2}-1} s_{n+\frac{N}{2}} W_N^{(n+\frac{N}{2})k} \\ &= \sum_{m=0}^{\frac{N}{2}-1} s_m W_N^{mk} + W_N^{\frac{N}{2}k} \sum_{n=0}^{\frac{N}{2}-1} s_{n+\frac{N}{2}} W_N^{nk} \\ S_k &= \sum_{m=0}^{\frac{N}{2}-1} s_m W_N^{mk} + W_r^{k} \sum_{n=0}^{\frac{N}{2}-1} s_{n+\frac{N}{2}} W_N^{nk} \end{aligned} \tag{3.3}$$

The final expression of S_k in Equation 3.3 consists of two summation of $\frac{N}{2}$ complex input terms scaled by complex exponential terms. There is an additional scalar complex exponential scaling applied to the 2^{nd} summation term.

The assumption made here is that all stages of computations share the same radix-r system that each multiplication and addition unit has $2r$ complex inputs in which r complex inputs are the data and r complex inputs are the complex exponential. The latency is $O(log_r(N))$ stages of complex block multiplication and addition. The computational complexity is on the order of $O(Nlog_r(N))$. A complex block floating-point format is also assumed on the input, intermediate buffering, and output of the FFT.

The input vector to the first stage will be multiplied by the 1^{st} stage complex coefficients, $W_N^k = 1, -j$ when $k = 0, \frac{N}{2}$, which is purely real and purely imaginary. In its implementation, the complex multiplication can be saved since complex multiplication by purely imaginary term is equivalent to phase shifting the complex operand by $\frac{\pi}{2}$. The N terms complex block addition is performed after the reordering of the 1^{st} stage complex input terms.

Figure 3.7 shows a 2-point FFT structure and bit packing of the input and output complex coefficients in the proposed block floating-point format. The twiddle factor is trivial which is a purely real number or purely imaginary number with unit magnitude. The 2-point FFT is the basic unit of structure in larger radix-2 FFT structure.



$\vec{x}$				
N_e	N_v	N_s	N_x	$N_m + 1$
Common Exponent	$\mathbb{R}\{x(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{x(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{R}\{x(1)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{x(1)\}$	Sign	Box Shift	Scaled Mantissa

$\vec{X}$				
N_e	N_v	N_s	N_x	$N_m + 1$
Common Exponent	$\mathbb{R}\{X(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{X(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{R}\{X(1)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{X(1)\}$	Sign	Box Shift	Scaled Mantissa

Figure 3.7: The butterfly unit of 2-point FFT is shown (twiddle factors omitted). The bit packing of complex input block and complex output block is shown for 2-point FFT.

In common exponent encoding/ exponent box encoding, a complex block of N_v samples can be decomposed into two complex block of $\frac{N_v}{2}$ samples each with the same common exponent. Since the common exponent is the same for two smaller complex block, the complex block addition operation that follows gets a free pre-processing of mantissa and common exponent prior to complex mantissa addition. In the worst case, the mantissa vector will get 1-bit wordlength expansion which might trigger increment of common exponent and truncation of complex mantissa vector. Since the output mantissa from the 1^{st} stage will be used for complex block multiplication and addition in the 2^{nd} stage, it may not be necessary to perform re-normalization immediately.

Figure 3.8 shows a 4-point FFT structure in the complex block floating-point format that has smaller structure 2-point FFT pre-processing. The common exponent from 1^{st} stage can propagate through to the $log_2(N)^{th}$ stage of FFT with minimal changes per stage due to the output of block addition of mantissa.

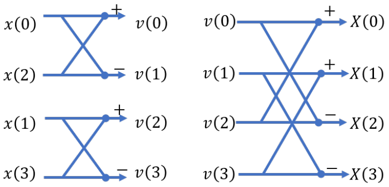
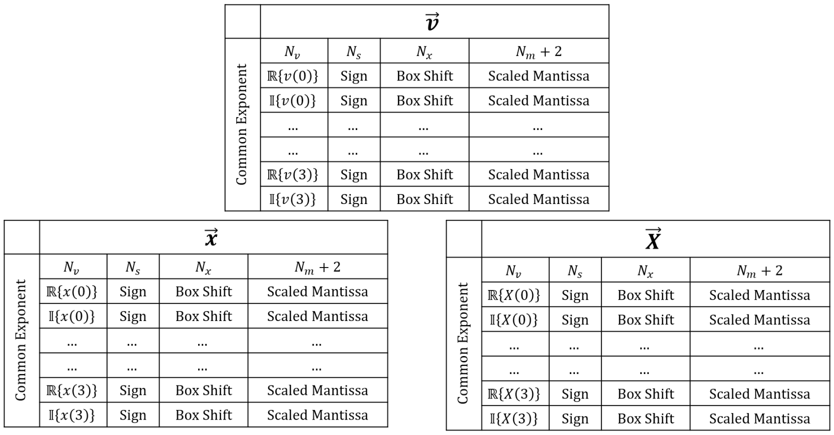


Figure 3.8: The butterfly unit of 4-point FFT in radix-2 format is shown (twiddle factors omitted). The bit packing of complex input block and complex output block is shown for 4-point FFT.

Figure 3.9 shows the bit packing of input, output complex coefficients and twiddle factor in the proposed block floating-point representation. The smallest non-zero twiddle factor is $W_4^1 = e^{-j\frac{2\pi}{4}1}$ that has exponent value of $+73$ in single-precision IEEE-754 format. The range of exponent values belonging to the twiddle factor is between $+73$ and $+127$. With sharing a common exponent per block, the second twiddle factor term, $W_4^1 = e^{-j\frac{2\pi}{4}1}$ is quantized to zero.



	$\vec{v}$			
Common Exponent	N_v	N_s	N_x	$N_m + 2$
	$\mathbb{R}\{v(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{v(0)\}$	Sign	Box Shift	Scaled Mantissa
	...	...	...	...
	...	...	...	...
	$\mathbb{R}\{v(3)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{v(3)\}$	Sign	Box Shift	Scaled Mantissa

	$\vec{x}$			
Common Exponent	N_v	N_s	N_x	$N_m + 2$
	$\mathbb{R}\{x(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{x(0)\}$	Sign	Box Shift	Scaled Mantissa
	...	...	...	...
	...	...	...	...
	$\mathbb{R}\{x(3)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{x(3)\}$	Sign	Box Shift	Scaled Mantissa

	$\vec{X}$			
Common Exponent	N_v	N_s	N_x	$N_m + 2$
	$\mathbb{R}\{X(0)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{X(0)\}$	Sign	Box Shift	Scaled Mantissa
	...	...	...	...
	...	...	...	...
	$\mathbb{R}\{X(3)\}$	Sign	Box Shift	Scaled Mantissa
	$\mathbb{I}\{X(3)\}$	Sign	Box Shift	Scaled Mantissa

Figure 3.9: The bit packing of complex input block, complex intermediate block, and complex output block is shown for 4-point FFT.

The radix-2 FFT processing unit can be easily scaled to perform larger point FFT such as 8-point FFT and 16-point FFT. Suppose the complex exponential term in the 2^{nd} stage are pre-computed and stored in look-up table (LUT), the number of non-trivial complex exponential terms are two terms. To allow for complex block arithmetic, the LUT may also store all complex exponential terms in the form of complex vectors that include trivial and non-trivial terms. The size of the LUT is Nr complex exponential terms.

3.3.2 Fast Fourier Transform (Radix-4)

With the consideration of computational latency, it is often preferred to compute fast Fourier transform in higher radices, such as radix-4 or radix-8 to obtain the result of fast Fourier transform in less number of stages. Each multiplication and addition unit will need r complex inputs in all stages. Suppose that N is a base-4 number. The Fourier transform formula can be rewritten in terms of four $N/4$-point FFT in Equation 3.4 with the following twiddle factors, $W_N = e^{-j\frac{2\pi}{N}}$ and radix number, $r = 4$.

$$
\begin{aligned}
S_k &= \sum_{m=0}^{N-1} s_m W_N^{mk} \\
&= \sum_{m=0}^{\frac{N}{4}-1} s_m W_N^{mk} + \sum_{n=\frac{N}{4}}^{\frac{N}{2}-1} s_n W_N^{nk} + \sum_{p=\frac{N}{2}}^{\frac{3N}{4}-1} s_p W_N^{pk} + \sum_{q=\frac{3N}{4}}^{N-1} s_q W_N^{qk} \\
&= \sum_{m=0}^{\frac{N}{4}-1} s_m W_N^{mk} + \sum_{n=0}^{\frac{N}{4}-1} s_{n+\frac{N}{4}} W_N^{(n+\frac{N}{4})k} + \sum_{p=0}^{\frac{N}{4}-1} s_{p+\frac{N}{2}} W_N^{(p+\frac{N}{2})k} + \sum_{q=0}^{\frac{N}{4}-1} s_{q+\frac{3N}{4}} W_N^{(q+\frac{3N}{4})k} \\
&= \sum_{m=0}^{\frac{N}{4}-1} s_m W_N^{mk} + W_N^{\frac{N}{4}k} \sum_{n=0}^{\frac{N}{4}-1} s_{n+\frac{N}{4}} W_N^{nk} + W_N^{\frac{N}{2}k} \sum_{p=0}^{\frac{N}{4}-1} s_{p+\frac{N}{2}} W_N^{pk} + W_N^{\frac{3N}{4}k} \sum_{q=0}^{\frac{N}{4}-1} s_{q+\frac{3N}{4}} W_N^{qk} \\
S_k &= \sum_{m=0}^{\frac{N}{4}-1} s_m W_N^{mk} + W_r^{k} \sum_{n=0}^{\frac{N}{4}-1} s_{n+\frac{N}{4}} W_N^{nk} + W_r^{2k} \sum_{p=0}^{\frac{N}{4}-1} s_{p+\frac{N}{2}} W_N^{pk} + W_r^{3k} \sum_{q=0}^{\frac{N}{4}-1} s_{q+\frac{3N}{4}} W_N^{qk}
\end{aligned}
\tag{3.4}
$$

Figure 3.10 shows the typical radix-4 butterfly unit that computes 4-point FFT in one stage. The amount of twiddle factors required per stage is r vectors with r complex exponentials per vector.

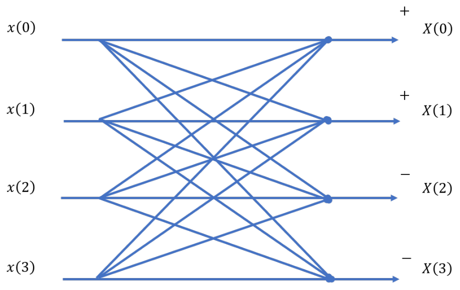


Figure 3.10: The butterfly unit of 4-point FFT in radix-4 is shown (twiddle factors omitted). The bit packing of complex input block and complex output block is shown for 4-point FFT.

3.4 Analysis and Limitation

3.4.1 Phase Resolution

The proposed representation of complex-valued pair gives 2x of B_m which has better effective phase resolution in quantization than that of common exponent representation. The inverse tangent function is a nonlinear function. The phase value of a complex number is obtained by fixing the real mantissa bits and increasing the imaginary bits that change the phase angle through the trigonometric function. Since common exponent is used in the complex block, the phase resolution observed at mantissa bits combination is independent of the actual exponent values. This implies that any complex value pair with magnitude coherence will have the same phase resolution that can be quantized in complex block representation.

Table 3.6 lists the phase resolution comparison of complex block representation for all IEEE-754 precisions. Figure 3.11 shows the phase resolution that can be quantized in the listed forms of complex block floating-point format. The exponent box encoding at 16-bit precision could achieve almost the same phase resolution achieved by common exponent encoding at 32-bit precision.

Mantissa Bits	**Common Exponent (rad)**	**Exponent Box (rad)**	**IEEE-754 (rad)**
11	4.88520×10^{-4}	2.38535×10^{-7}	9.31778×10^{-10}
24	5.96046×10^{-8}	3.55271×10^{-15}	2.80260×10^{-45}
53	1.11022×10^{-16}	1.23260×10^{-32}	4.94066×10^{-324}
113	9.62964×10^{-35}	9.27302×10^{-69}	undefined

Table 3.6: The phase resolution means the minimum quantized phase change. The phase resolution is measured by computing the inverse tangent function of the ratio of imaginary amplitude and the real amplitude.

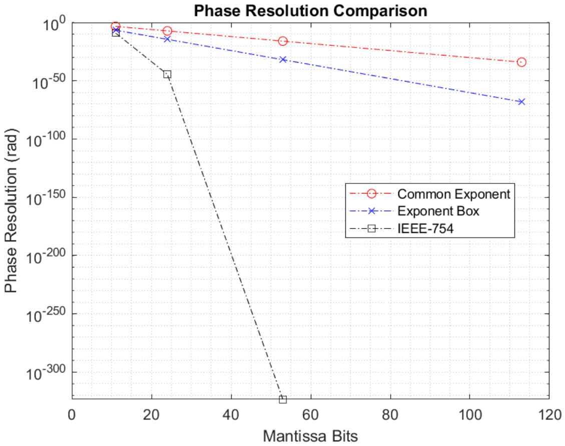


Figure 3.11: The plot of phase resolution compares among the combination of mantissa bits and complex block floating-point format.

3.4.2 Block Wordlength Analysis

Table 3.7 summarizes the block wordlength requirement to represent N_v complex samples in memory or in vector floating-point register.

Encoding Method	Block Wordlength (bits)	Bit Savings
Complex IEEE754	$2N_v(N_s + N_e + N_m)$	0
Common Exponent	$2N_v(N_s + N_m^*) + N_e$	$N_e - 2 + (2N_e - 2)(N_v - 1)$
Exponent Box	$2N_v(N_s + N_m^* + N_x) + N_e$	$N_e - 4 + (2N_e - 4)(N_v - 1)$

Table 3.7: The wordlength requirement per complex block is defined in terms of N_v complex samples per block. The N_m^* for common exponent and exponent box format refers to N_m+1 for the complex IEEE754 format.

3.4.3 Block Arithmetic Complexity Analysis

Table 3.8 lists the comparison of computational complexity in the complex block addition arithmetic. Table 3.9 lists the comparison of computational complexity in the complex block multiplication arithmetic. The number of hardware operations involve the shifters, adders, comparator, multipliers, and 2's complement logic.

Hardware Units	**Complex IEEE754**	**Common Exponent**	**Exponent Box**
Shifters	$4N_v$	$4N_v$	$12N_v$
$(N_m + 1)$-bit Adders	$2N_v$	$2N_v$	$2N_v$
N_e-bit Adders	$3N_v$	2	5
N_e-bit Comparator	N_v	1	1
$(N_m + 1)$-bit 2's Complement	$4N_v$	$4N_v$	$4N_v$
1-bit Comparator	0	0	$4N_v$

Table 3.8: The complexity analysis for implementing complex block addition is determined in terms of type and frequency of hardware operations and corresponding bit widths. N_v is number of complex samples per block.

For the complex block addition, the direct comparison shows that common exponent and exponent box encoding have reduced operations on the number of N_e-bit addition and comparators. The number of shifters and 1-bit comparators needed in exponent box encoding increases.

Hardware Units	**Complex IEEE754**	**Common Exponent**	**Exponent Box**
Shifters	$4N_v$	$2N_v$	$4N_v$
$(N_m + 1)$-bit Multipliers	$4N_v$	$4N_v$	$4N_v$
N_e-bit Adders	$14N_v$	3	4
N_e-bit Comparator	$2N_v$	0	$2N_v$
$(N_m + 1)$-bit 2's Complement	$4N_v$	$4N_v$	$4N_v$
1-bit XOR Unit	$4N_v$	$4N_v$	$4N_v$
1-bit Adders	0	0	$4N_v$
2-bit Comparator	0	0	$4N_v$

Table 3.9: The complexity analysis for implementing complex block multiplication is determined in terms of type and frequency of hardware operations and corresponding bit widths. N_v is number of complex samples per block.

For the complex block multiplication, the direct comparison shows that common exponent has reduced operations on the number of N_e-bit addition and comparators. The exponent box encoding has only reduced operations on N_e-bit adders but increased operations on 1-bit adders and 2-bit comparators.

Chapter 4

System Model of Baseband Transceivers Design

4.1 Single-carrier Modulation System

In single-carrier modulation system, the carrier signal with frequency, f_c, carries two independent streams of baseband symbols if quadrature amplitude modulation (QAM) is used. The baseband symbols are known as in-phase waveform, $I(t)$ and quadrature-phase waveform, $Q(t)$ which mean the phase difference relative to carrier frequency. M-QAM consists of a total of M symbols in the constellation set. The value of M is a base-2 number and $J = log_2(M)$ is the number of information bits encoded by each symbol. To achieve high data rates, one of the two following options is possible. The first option is to pick a large symbol rate, f_{sym}, however, higher sampling rate, f_s may be required since $f_s > f_{sym}$. The other option is to make sure the number of baseband symbol per constellation, M to be high. Figure 4.1 shows an example 256-QAM normalized constellation.

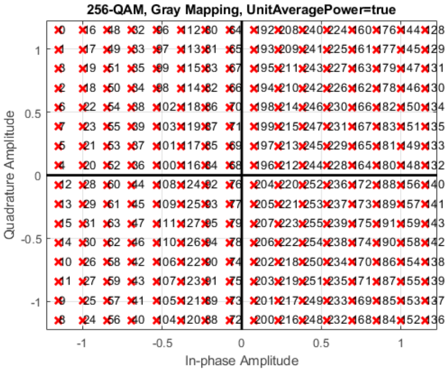


Figure 4.1: The example plot of normalized energy 256-quadrature amplitude modulation (QAM) constellation.

4.1.1 Discrete-time Complex Baseband QAM Transmitter

Figure 4.2 describes the block diagram of a typical baseband QAM transmitter. The information bit stream is generated at the rate of Jf_{sym} and is fed into the symbol mapper. The symbol mapper selects from a set of M-QAM complex symbols at a rate of f_{sym}. The symbol mapper performs conversion from information in logical bits to complex symbols in numeric quantization bits. The upsampler block has a memory of L complex samples, takes input at rate of f_{sym}, and generates output at rate of $f_s = Lf_{sym}$. However, the non-trivial output values are generated at rate of f_{sym} and the rest are simply zeros.

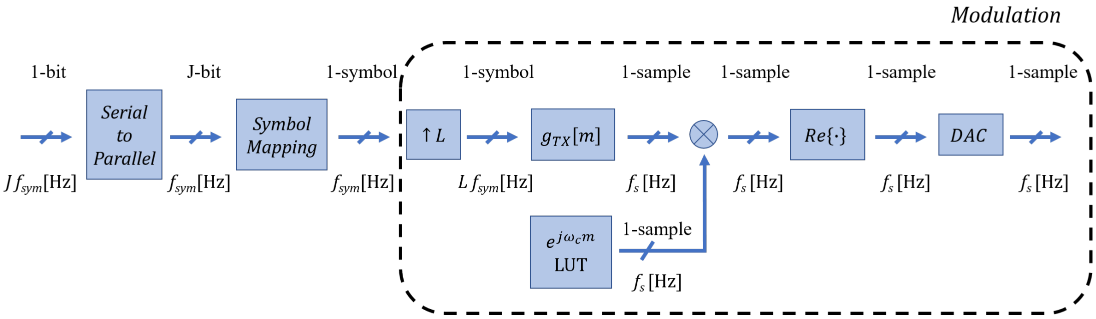


Figure 4.2: The typical block diagram of complex baseband QAM transmitter consists of conversion from logical bits to analog waveform. The complex block size can be varied between each functional block.

In the case of single-carrier QAM transmitter, the sampling frequency has the following expression, $f_s = Lf_{sym}$ Hz where L is the upsample factor and f_{sym} is the complex baseband symbol rate. Table 4.1 lists the memory read rates (complex samples per second), memory write rates (complex samples per second), and multiply-accumulate rate (complex multiply-add per second) required for implementing the baseline baseband M-QAM transmitter processing chain. There may be additional processing overheads required for handling buffering mechanism.

With complex block representation, the above complexity analysis will be

Transmitter Chain	Memory read Rate (complex samples/sec)	Memory write Rate (complex samples/sec)	Multiply-Accumulate Rate (COMPLEX MULT-ADD/sec)
Sym Mapper	Jf_{sym} [bps]	f_{sym}	0
Upsampler	f_{sym}	f_s	0
RRC Filter (k_{rrc}^{th} order)	$2(k_{rrc}+1)f_s$	f_s	$(k_{rrc}+1)f_s$ COMP. MULT $k_{rrc}f_s$ COMP. ADD
Upconvert	$2f_s$	f_s	f_s COMP. MULT

Table 4.1: The memory access complexity of complex baseband QAM transmitter is measured in terms of memory read/write rate and the computational complexity is measured in terms of multiply-accumulate rate.

slightly different. For all functional units in the transmitter processing chain, N_v is defined as the number of complex samples in a block and f_b is defined as the block rate of data samples feeding into the functional units. $N_v(i)f_b(i) = f_{read}(i)$ is the constraint on meeting the memory read rates of functional unit i. To be consistent with definition defined in Table 3.7, Section 3.4.2, $N_w(i)$ is defined as the block wordlength (bits) to represent $N_v(i)$ complex samples in a block for functional unit i.

4.1.2 Discrete-time Complex Baseband QAM Receiver

Figure 4.3 describes the block diagram of a typical baseband QAM receiver. The equalizer has input rates of f_s and output rates of f_s. If the equalizer is implemented as N_{eq}^{th}-order FIR filter, it has $N_{eq}^{th}+1$ coefficients in memory. The demodulator is complex block multiply with input rates of f_s and output rates of f_s. The matched filter has input and output rates of f_s and the matched filter is N_{mf}^{th}-order where $N_{mf} = L_{RX}N_g$, L_{RX} is the upsample factor at the receiver, and N_g is the span of symbols in the filter. The downsampler has input rates of f_s and output rates of f_{sym}. The symbol demapper has input rates of f_{sym} complex samples per second and output rates of Jf_{sym} bits per second.

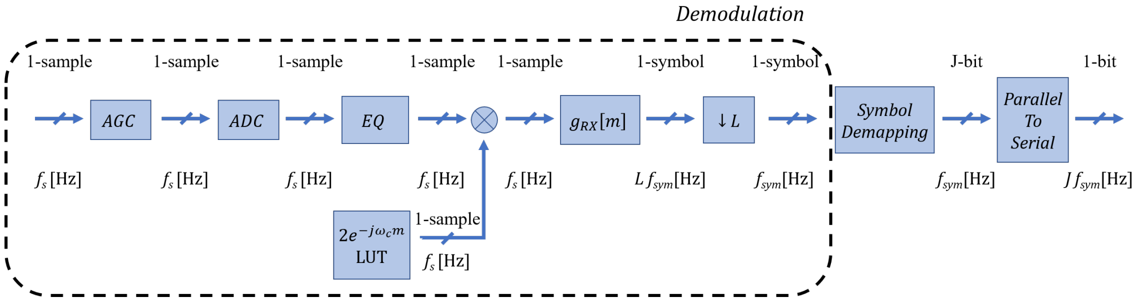


Figure 4.3: The typical block diagram of complex baseband QAM receiver consists of front end processing of data conversion from analog waveform to logical bits. The complex block size can be varied between each functional block.

Table 4.2 lists the memory read rates (complex samples per second), memory write rates (complex samples per second), and multiply-accumulate rate (complex multiply-add per second) required for implementing the baseline baseband M-QAM receiver processing chain.

Receiver Chain	Memory read Rate (complex samples/sec)	Memory write Rate (complex samples/sec)	Multiply-Accumulate Rate (COMPLEX MULT-ADD/sec)
Equalizer (k_{eq}^{th} order)	$2(k_{eq}+1)f_s$	f_s	$(k_{eq}+1)f_s$ COMP. MULT $k_{eq}f_s$ COMP. ADD
Downconvert	$2f_s$	f_s	f_s COMP. MULT
Matched Filter (k_{mf}^{th} order)	$2(k_{mf}+1)f_s$	f_s	$(k_{mf}+1)f_s$ COMP. MULT $k_{mf}f_s$ COMP. ADD
Downsampler	f_s	f_{sym}	0
Sym. Demapper	f_{sym}	$Jf_{sym}[bps]$	0

Table 4.2: The memory access complexity of complex baseband QAM transmitter is measured in terms of memory read/write rate and the computational complexity is measured in terms of multiply-accumulate rate.

For all functional units in the receiver processing chain, N_v is defined as the number of complex samples in a block and f_b is defined as the block rate of data samples feeding into the functional units. $N_v(i)f_b(i) = f_{read}(i)$ is the constraint on meeting the memory read rates of functional unit i. To be consistent with definition defined in Table 3.7, Section 3.4.2, $N_w(i)$ is defined as the block wordlength (bits) to represent $N_v(i)$ complex samples in a block for functional unit i.

4.1.3 Channel Model and Assumptions

The channel model assumed in the system model is Additive White Gaussian Noise (AWGN).

4.1.4 LTE Specifications and Requirements

Table 4.3 contains the parameter definitions and values that describes the QAM constellations. Table 4.4 contains the specifications for QAM transmitter and QAM receiver specifications in the system model. The specifications used in this simulation are influenced by the requirement of Long Term Evolution (LTE) base station, Release 12.

QAM Parameters	Definition	Values / Types
Constellation Order	M	{4, 16, 64, 256}
Constellation Shape	-	Square
Normalized	-	Yes

Table 4.3: The QAM specifications include the constellation order, shape, and normalization that the transmitter and receiver will follow.

The error vector magnitude (EVM) is defined as the square root of ratio of mean error vector power of the measured symbols to the mean reference power of the ideal symbols in percentage. The EVM requirement is specified for the transmitter conformance testing that depends on transmit symbol constellation. Table 4.5 lists the EVM requirement of the transmitter specified in LTE base station design.

The adjacent channel leakage ratio (ACLR) is defined as the ratio of filtered mean power centered on the assigned channel frequency to the filtered mean power

Transceiver Parameters	Definition	Values / Types
Filter Bandwidth (MHz)	B	5, 10, 15, 20
Up-sample Factor	L_{TX}, L_{RX}	5, 10, 15, 20
Symbol Rate (MHz)	f_{sym}	1
Symbol Periods	N_g	8
Filter Order	N_{TX}, N_{RX}	$\{40^{th}, 80^{th}, 120^{th}, 160^{th}\}$
Pulse Shape	g_{TX}, g_{RX}	Root-Raised Cosine
Excess Bandwidth Factor	α_{TX}, α_{RX}	0.4

Table 4.4: The implementation specifications of digital transmitter and receiver have balanced the tradeoffs between cost and performance.

Constellation Size	(4-QAM)	(16-QAM)	(64-QAM)	(256-QAM)
EVM	18.5 %	13.5 %	9 %	unspecified

Table 4.5: The error vector magnitude (EVM) requirement is listed as of specification in the Long-Term Evolution (LTE) base station standards.

centered on adjacent channel frequency. Table 4.6 lists the ACLR requirement of transmitter based on operation in unpaired spectrum.

Adjacent Channel Frequency	Adjacent Channel Carrier [Mcps]	Filter Bandwidth [Mcps]	ACLR Limit
BW/2+0.8MHz	1.28	RRC (1.28)	45 dB
BW/2+2.4MHz	1.28	RRC (1.28)	45 dB
BW/2+7.5MHz	3.84	RRC (3.84)	45 dB
BW/2+5MHz	7.68	RRC (7.68)	45 dB
BW/2+15MHz	7.68	RRC (7.68)	45 dB

Table 4.6: The adjacent channel leakage ratio (ACLR) requirement is listed as of specification in the Long-Term Evolution (LTE) base station standards with channel bandwidth 5, 10, 15, and 20 MHz.

4.2 Multi-carrier Modulation System

Multi-carrier modulation system divides a carrier signal with large bandwidth, B Hz into N_f subcarrier signals with smaller bandwidth, $\frac{B}{N_f}$ Hz. Each subcarrier carries either a baseband symbol or no information. The main motivation of using multi-carrier modulation system is that a frequency selective wideband channel can be reduced to multiple frequency flat narrowband channels. As a consequence, with the estimated channel coefficients, the channel equalization operation at the receiver can be reduced to $O(N_f)$ vector-based division, where N_f is the number of subcarriers. In terms of implementation, fast Fourier transform based structure is more efficient than matrix based structure. The orthogonal frequency division modulation (OFDM) system is based on fast Fourier transform approach. The periodicity in OFDM transmit symbol is created by prepending a portion of the tail of the OFDM symbol to the head of the OFDM symbol.

Each OFDM symbol has N_f complex baseband QAM symbols in a block, which determines the block size of complex block and block wordlength defined in Table 3.7, Section 3.4.2. Each OFDM symbol undergoes $log_r(N_f)$ stages of r-point multiply-accumulate for each frequency-time (IFFT) and time-frequency (FFT) conversion, where r is the radix number of FFT and IFFT Butterfly Unit. The complex block arithmetic unit through the cascaded stage of FFT and IFFT units would generate wordlength expansion at each stage and rescaling of the shared exponent.

4.2.1 Discrete-time OFDM Transmitter

Figure 4.4 shows the block diagram of OFDM transmitter. When there is only a subset of N_f subcarriers assigned for carrying data and control symbols, i.e. $N_f - K$ are data and control complex symbols, then there are K null tones inserted in the OFDM modulation. This is also beneficial when channel estimation feedback provides knowledge that certain subcarriers, s_i have weak complex channel gains. The cyclic prefix is intentionally added in the OFDM transmitter for the purpose of creating periodicity in modulated data symbols. The number of cyclic prefix assigned have to be greater than inter-symbol interference.

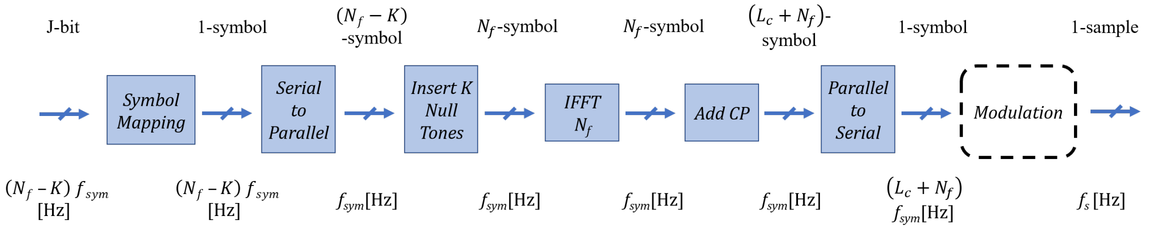


Figure 4.4: The typical block diagram of orthogonal frequency division multiplexing (OFDM) transmitter. The block size of each functional unit can be chosen to be N_f, the number of subcarriers.

In the case of OFDM transmitter, the sampling frequency has the following expression, $f_s = (L_c + N_f)f_{sym}$ Hz, where L_c is the length of cyclic prefix, N_f is the number of subcarriers per OFDM symbol, and f_{sym} is the complex baseband symbol rate. Table 4.7 lists the complexity analysis of OFDM transmitter.

Transmitter Chain	Memory read Rate (complex samples/sec)	Memory write Rate (complex samples/sec)	Multiply-Accumulate Rate (COMPLEX MULT-ADD/sec)
Sym Mapper	$J(N_f - K)f_{sym}$	$(N_f - K)f_{sym}$	0
Insert K Null Tones	$(N_f - K)f_{sym}$	$N_f f_{sym}$	0
Inverse FFT	$N_f f_{sym}$	$N_f f_{sym}$	$N_f log_r(N_f)f_{sym}$ C. MULT $(N_f - 1)log_r(N_f)f_{sym}$ C. ADD
Insert Cyclic Prefix	$N_f f_{sym}$	$(L_c + N_f)f_{sym}$	0
Modulation	$(L_c + N_f)f_{sym}$	f_s	f_s COMP. MULT

Table 4.7: The memory access complexity of OFDM transmitter is measured in terms of memory read/write rate and the computational complexity is measured in terms of multiply-accumulate rate.

4.2.2 Discrete-time OFDM Receiver

Figure 4.5 shows the block diagram of OFDM receiver.

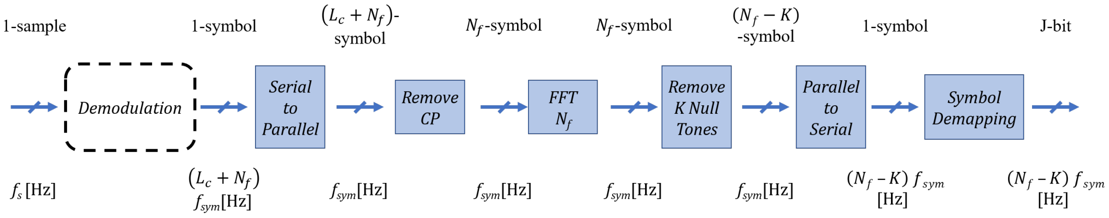


Figure 4.5: The typical block diagram of orthogonal frequency division multiplexing (OFDM) receiver. The block size of each functional unit can be chosen to be N_f, the number of subcarriers.

Table 4.8 lists the complexity analysis of OFDM receiver.

Receiver Chain	Memory read Rate (complex samples/sec)	Memory write Rate (complex samples/sec)	Multiply-Accumulate Rate (COMPLEX MULT-ADD/sec)
Demodulation	$2f_s$	$(L_c + N_f)f_{sym}$	f_s COMP. MULT
Remove Cyclic Prefix	$(L_c + N_f)f_{sym}$	$(N_f)f_{sym}$	0
FFT	$(N_f)f_{sym}$	$(N_f)f_{sym}$	$N_f log_r(N_f)f_{sym}$ C. MULT $(N_f - 1)log_r(N_f)f_{sym}$ C. ADD
Remove K Null Tones	$(N_f)f_{sym}$	$(N_f - K)f_{sym}$	0
Symbol Demapper	$(N_f - K)f_{sym}$	$J(N_f - K)f_{sym}$ [bps]	0

Table 4.8: The memory access complexity of OFDM receiver is measured in terms of memory read/write rate and the computational complexity is measured in terms of multiply-accumulate rate.

4.2.3 Channel Model and Assumptions

The channel model is assumed to be $L^{th}_{channel}$-order.

4.2.4 LTE Specifications and Requirements

Table 4.9 specifies the additional requirement on EVM window length.

Channel Bandwidth [MHz]	FFT Size	CP Length Sym 0	CP Length Sym 1-6	EVM Window Length
1.4	128	10	9	5
3	256	20	18	12
5	512	40	36	32
10	1024	80	72	66
15	1536	120	108	102
20	2048	160	144	136

Table 4.9: The error vector magnitude (EVM) window length of OFDM transmitter as of the long-term evolution (LTE) base station standards.

Chapter 5

Desktop Simulation

5.1 Signals Generation

The signals generated in this simulation are complex normal random vector according to complex normal distribution specified in Equation 5.1 and $Z = X + jY$. The generated signals are quantized to 32-bit single precision floating-point number for both inputs of complex block arithmetic unit. Under normal operating condition, the common exponent encoding works well.

$$\begin{aligned} \varphi(t) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} \\ X &\sim \varphi(t) \\ Y &\sim \varphi(t) \end{aligned} \tag{5.1}$$

The exponent box encoding would perform better in practical operating condition when the input signals to the complex block arithmetic unit are generated from multiple sources. With the interfering signals condition, the input signals could be modeled as linear combination of two normally distributed random vectors with different statistics (mean and variance). With the additive white noise condition, the input signals may have a difference distribution from the noise distribution. The current simulation explores both conditions.

The simulation software is published under MIT license at Github repository [11].

5.2 MATLAB: Vector Arithmetic Unit

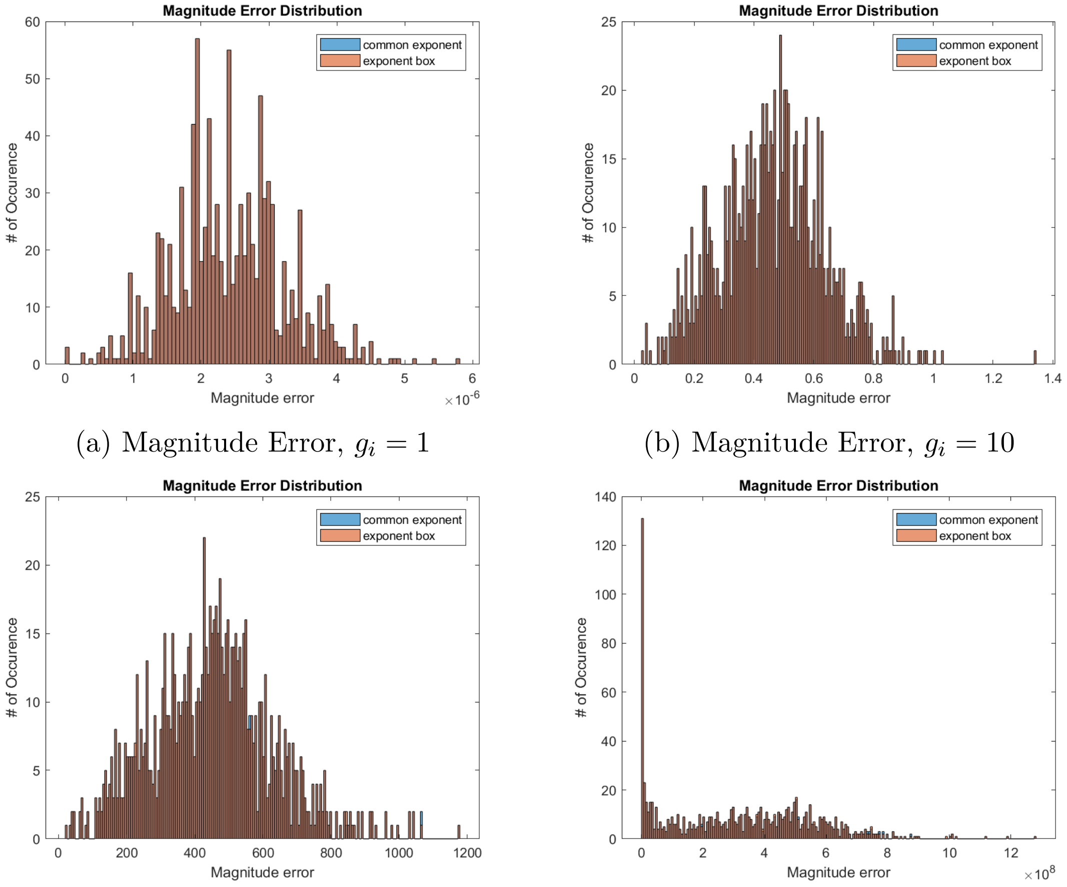
Complex block addition is not the main focus of this work. The exponent box encoding format would provide lower quantization error. However, part of the complex block addition will reverse the process of exponent box encoding. Therefore, the simulation of complex block addition in common exponent encoding and exponent box encoding will give the same results. The simulation of complex block addition is currently omitted in this section.

Complex block multiply is implemented for 32-bit precision. Each multiply has two input blocks with $N_v = 1,000$ complex samples per block. The inputs to the complex block multiply are modified Gaussian where they are initially generated with normal distribution and unit variance. In each complex block, one input term, i^{th} is randomly selected to be applied with additional gain, g_i.

Figure 5.1 shows the magnitude error distribution of the multiply result when the additional gain, $g_i = 2^1, 2^{10}, 2^{15}, 2^{25}$. For 32-bit precision, the mantissa are 23 bits which corresponds to 8×10^6 in magnitude difference. Each bit error corresponds to 10^{-6} of the exponent value. As gain value, g_i increases by 10x, the span of magnitude error also corresponds to 10^{-6} of the new gain value.

Figure 5.2 shows the phase error distribution of the multiply result when the additional gain, $g_i = 2^1, 2^{10}, 2^{15}, 2^{25}$. The phase error distribution shows an interesting finding that 4 different clusters of about the same size. None of the error distribution plots show that the bin that contains phase error of 0 is the majority. The important takeaway is the due to sharing a common exponent in complex block, the phase resolution of each sample starts to degrade. For $g_i = 25$, it begins to show that the error bin containing the zero phase error shows a peak.

Figure 5.3 shows the scatter plot of complex value pairs of the multiply re-



(a) Magnitude Error, $g_i = 1$

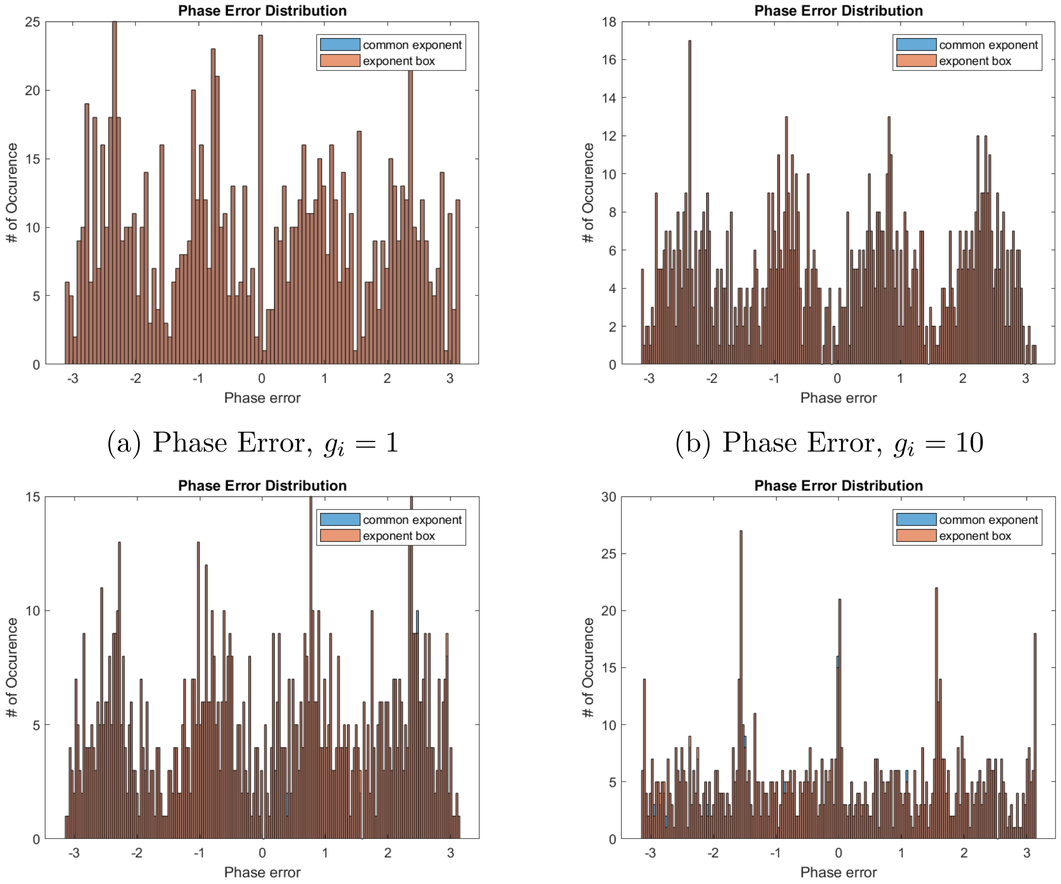
(b) Magnitude Error, $g_i = 10$

(c) Magnitude Error, $g_i = 15$

(d) Magnitude Error, $g_i = 25$

Figure 5.1: The complex block inputs are initially generated from normal distribution with unit variance and quantized to 32-bit floating-point number per dimension. A fixed gain value, g_i is applied on a randomly selected term of the complex input block. The magnitude error distributions are plotted for corresponding gain values, g_i.

sult when the additional gain, $g_i = 2^1, 2^{10}, 2^{15}, 2^{25}$. The scatter plot is obtained by normalizing each dimension of complex pair by the maximum absolute value per dimension. When $g_i = 1$, the scatter plot shows a Gaussian distribution where the top right corner has the largest value. When $g_i = 10$, the scatter plot shows a non-linear transformed version of Gaussian distribution towards the bottom left corner. That can be explained by the scaled mantissa vector in the complex block has been scaled due to sharing of common exponent. When $g_i = 15$, the scaling of mantissa



(a) Phase Error, $g_i = 1$

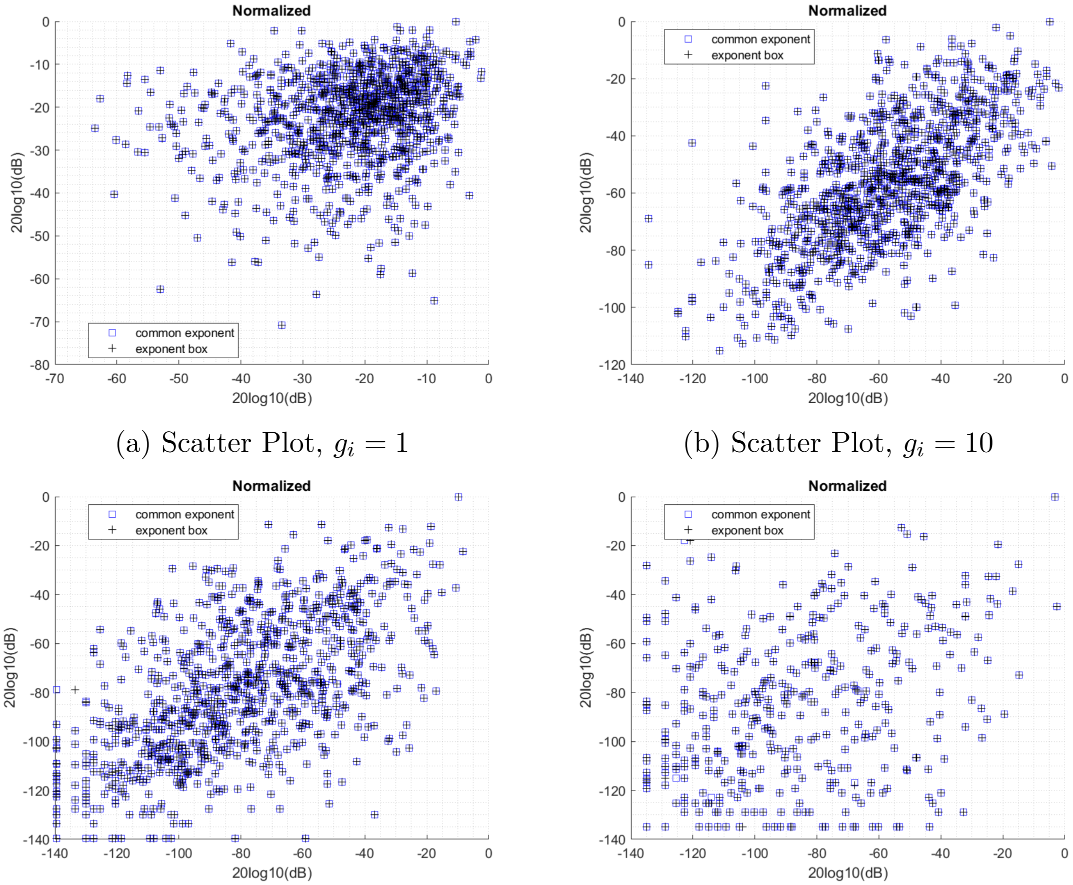
(b) Phase Error, $g_i = 10$

(c) Phase Error, $g_i = 15$

(d) Phase Error, $g_i = 25$

Figure 5.2: The complex block inputs are initially generated from normal distribution with unit variance and quantized to 32-bit floating-point number per dimension. A fixed gain value, g_i is applied on a randomly selected term of the complex input block. The phase error distributions are plotted for corresponding gain values, g_i.

vector continues and some of the complex value pair with smaller exponent values start hitting the boundary of effective encoding region determined by the bit width of mantissa. That can be observed with a number of complex pairs lie on the left (real mantissa becomes zero) and bottom (imaginary mantissa becomes zero) of the complex plane. When $g_i = 25$, the exponent rescaling is the most aggressive. There are the most complex pairs that lie on the left (real mantissa becomes zero) and bottom (imaginary mantissa becomes zero) of the complex plane.



(a) Scatter Plot, $g_i = 1$

(b) Scatter Plot, $g_i = 10$

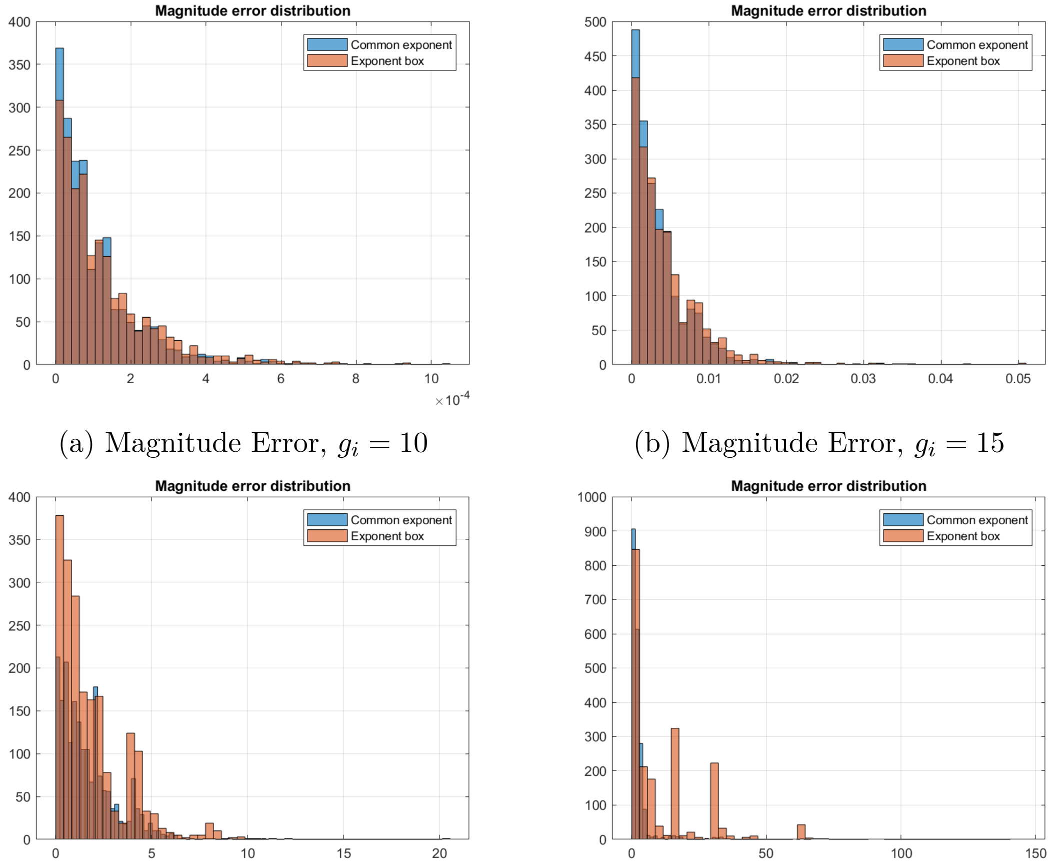
(c) Scatter Plot, $g_i = 15$

(d) Scatter Plot, $g_i = 25$

Figure 5.3: The complex block inputs are initially generated from normal distribution with unit variance and quantized to 32-bit floating-point number per dimension. A fixed gain value, g_i is applied on a randomly selected term of the complex input block. With increasing gain values, a larger dynamic range is required.

5.3 MATLAB: Algorithms Modeling Unit

Fast fourier transform algorithm is implemented in complex block floating-point format. Higher radices such as $r = 4$ and $r = 8$ are more commonly used when the number of samples per vector is high, $N = 1024$ and $N = 2048$. With radix-4 implementation, Figure 5.4 shows the magnitude error distribution of 4-point FFT output coefficients.



(a) Magnitude Error, $g_i = 10$

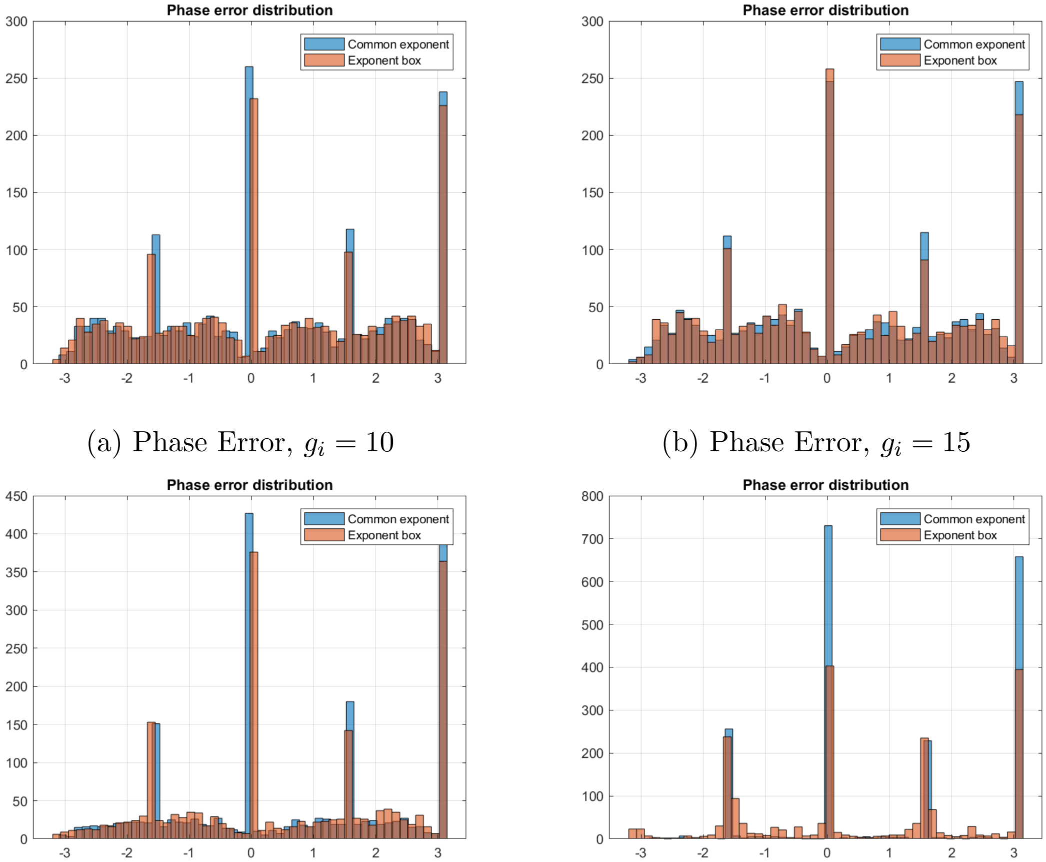
(b) Magnitude Error, $g_i = 15$

(c) Magnitude Error, $g_i = 25$

(d) Magnitude Error, $g_i = 28$

Figure 5.4: The 4-point fast Fourier transform (FFT) in radix-4 implementation has complex block inputs generated from normal distribution with unit variance. A randomly selected term of the complex block input is added with additional gain, g_i specified. The magnitude error distribution is plotted for each g_i.

Figure 5.5 shows the phase error distribution of 4-point FFT output coefficients.



(a) Phase Error, $g_i = 10$

(b) Phase Error, $g_i = 15$

(c) Phase Error, $g_i = 25$

(d) Phase Error, $g_i = 28$

Figure 5.5: The 4-point fast Fourier transform (FFT) in radix-4 implementation has complex block inputs generated from normal distribution with unit variance. A randomly selected term of the complex block input is added with additional gain, g_i specified. The phase error distribution is plotted for each g_i.

Chapter 6

Conclusion

6.1 Summary

This research work investigates the possibility of using complex block floating-point format as the standard floating-point format for vector-based data. The proposed complex block floating-point format is derived from IEEE-754 format and extends to two dimensional data that have sensitivity in magnitude and phase. The main assumption used in this proposed format is the coherence in magnitude and phase across multiple data point in a complex valued vector.

Chapter 1 motivates the design of hardware structure in the form of complex block floating-point format for implementation of digital communication systems. The quadrature amplitude modulation based communication systems typically has complex baseband symbol in vector form for baseband transmitter and baseband receiver processing.

Chapter 2 reviews on the standard IEEE-754 floating-point format because the its abundant implementation of single-precision (32-bit) and double-precision (64-bit) in modern processor and arithmetic units. There are also usage of half-precision (16-bit) floating-point format in transmitter and receiver design. The discussion of standard IEEE-754 floating-point format extends to complex-valued representation (2 dimensional scalar data of similar precision) and complex block floating-point format (2 dimensional vector data of similar precision).

Chapter 3 describes a proposed complex block floating-point format that has the basis of block floating-point (shared exponent across the entire complex vector) with extension of dynamic range using two additional bit per complex sample in the block. The vector arithmetic subsystem is modeled for complex block addition and complex block multiplication on the proposed data type. The fast Fourier transform algorithm is evaluated for radix-2 and radix-4 type in the proposed representation.

Chapter 4 evaluates the transmitter and receiver design in the single-carrier and multi-carrier modulation system in the form of proposed complex block floating-point format. The main specification of the transceiver design is obtained from the Long-Term Evolution (LTE) Base Station Standard, Release 12.

Chapter 5 concludes with the numeric and graphical results of the simulation performed in MATLAB environment. The major pieces of MATLAB simulation are conducted on evaluating the performance of complex block arithmetic and algorithm modeling unit in the proposed complex block representation. The transmitter and receiver design simulator are also simulated for single-carrier modulation system and multi-carrier modulation system.

6.2 Future Work

The simulation results obtained thus far from this research project look encouraging and the idea of using complex block floating-point for the standard data type in vector based processor looks promising. For future work extending from here, there are a few main directions that seem possible.

For hardware design and implementation of complex block arithmetic, the block diagrams used in complex block addition and multiplication have not been heavily optimized. The current model of complex block arithmetic shows the base model in which all bits used in representing the proposed complex block format have been meaningfully used for computation of the block arithmetic results.

For specific design of receiver baseband processing, it maybe possible to optimize hardware energy requirement by allowing multi-level precision in receiver system model such that the highest precision quantization bits is required at the ADC end and decreases to the lowest precision quantization bits at the symbol de-mapper end. The modeled communication system in current work is single-input single-output (SISO) channel model. The work can be extended to multi-input multi-output (MIMO) channel model for more relevance in the communication system implementation in practice today.

Appendices

Appendix A

Maximum Exponent Difference for Low Quantization Error

Let i, j be two bounded positive real numbers represented in floating point precision. Assume that i has larger magnitude than j, such that $|j| < |i|$. Define $E(k)$ as exponent of k, $M(k)$ as mantissa of k, and $B(k) = 2^{E(k)-1}-1$ as exponent bias, where $k = \{i, j\}$. It is assumed that the bias of two floating point, i, j, is the same, i.e. $B(i) == B(j)$. Let the difference between two exponents be $\Delta E = E(i) - E(j) > 0$.

Since it is assumed that $|j| < |i|$,

$$
\begin{aligned}
|j| &< |i| \\
(1.M(j) * 2^{E(j)-B(j)}) &< (1.M(i) * 2^{E(i)-B(i)}) \\
(1.M(j) * 2^{E(j)}) &< (1.M(i) * 2^{E(i)}) \\
(1.M(j) * 2^{E(j)-E(i)+E(i)}) &< (1.M(i) * 2^{E(i)}) \\
(1.M(j) * 2^{E(j)-E(i)}) &< (1.M(i)) \\
(1.M(j) * 2^{-\Delta E}) &< (1.M(i)) \\
(0.M(j')) &< (1.M(i)) \\
where \quad M(j') &= \frac{1.M(j)}{2^{\Delta E}}
\end{aligned}
\tag{A.1}
$$

The mantissa bits in $M(j')$ are truncated in practice, therefore, ΔE must be less than $M(j)$ for low quantization error. The quantization error is the largest when the $M(j')$ gets zero and $M(j)$ is nonzero.

Appendix B

Phase Resolution in Common Exponent Encoding

Suppose a complex floating-point vector, $\boldsymbol{X}$ has the numeric precision defined in Section 2.2.1 and 3.1. The complex vector, $\boldsymbol{X}$ is defined as $\boldsymbol{X} = \boldsymbol{X}_{Real} + j\boldsymbol{X}_{Imag}$. The magnitude and phase vector representing the vector $\boldsymbol{X}$ can be defined as, $|\boldsymbol{X}| = \sqrt{\boldsymbol{X}_{Real}^2 + \boldsymbol{X}_{Imag}^2}$ and $\theta_{\boldsymbol{X}} = tan^{-1}(\frac{\boldsymbol{X}_{Imag}}{\boldsymbol{X}_{Real}})$.

The smallest phase increment has the form of the following equations when the difference and arc tan function are reordered,

$$\begin{aligned} \Delta\ \theta &= \Delta\ tan^{-1}(\frac{X_{Imag}}{X_{Real}}) \\ &\approx tan^{-1}(\frac{\Delta\ X_{Imag}}{X_{Real}}) \end{aligned} \tag{B.1}$$

Since $\boldsymbol{X}_{Imag}$ and $\boldsymbol{X}_{Real}$ share the same exponent in a complex block floating-point format, they have the form, $\boldsymbol{X}_{Imag} = (-1)^{\boldsymbol{S}_{Imag}}\boldsymbol{M}_{Imag}2^{E_{block}}$ and $\boldsymbol{X}_{Real} = (-1)^{\boldsymbol{S}_{Real}}\boldsymbol{M}_{Real}2^{E_{block}}$. Rewriting the above equation, we have,

$$\begin{aligned} \Delta\ \theta &\approx tan^{-1}(\frac{\Delta\ X_{Imag}}{X_{Real}}) \\ min_{\Delta X_{Imag}}(\Delta\ \theta) &\approx tan^{-1}(\frac{0.0...1}{M_{Real}}) \end{aligned} \tag{B.2}$$

The smallest phase increment is obtained by maximizing the mantissa values of the real part in the complex block. The phase difference in the complex pair is a function of real mantissa value and increment of imaginary mantissa value. The phase resolution is independent of the exponent value.

Appendix C

Block Diagrams for Complex Block Arithmetic

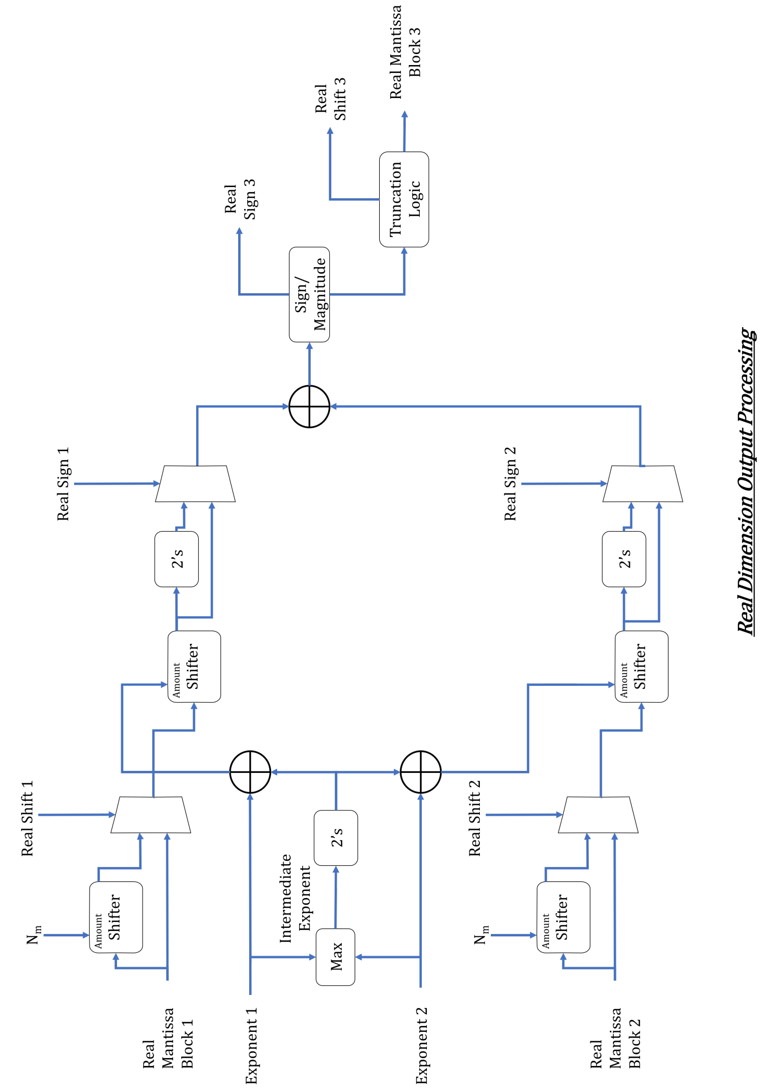


Figure C.1: Complex Block Addition Block Diagram (i) Real Output

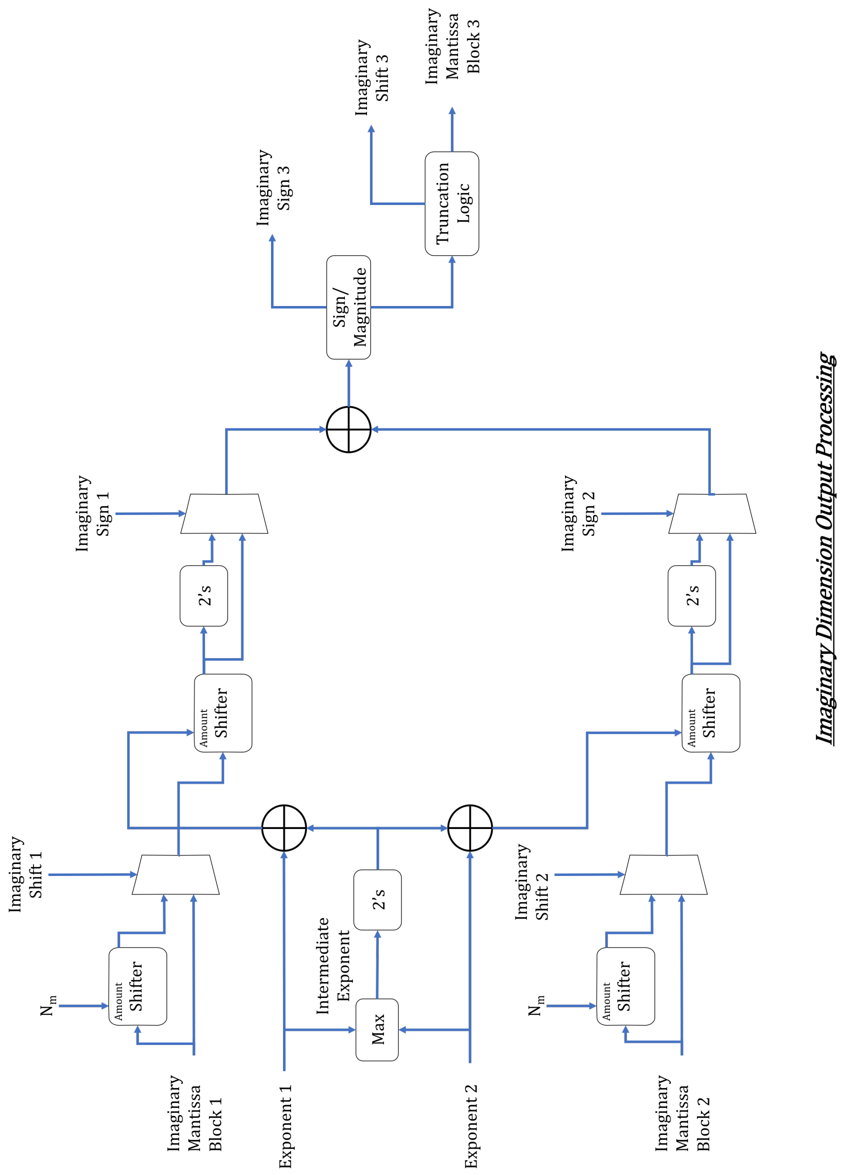


Figure C.2: Complex Block Addition Block Diagram (ii) Imaginary Output

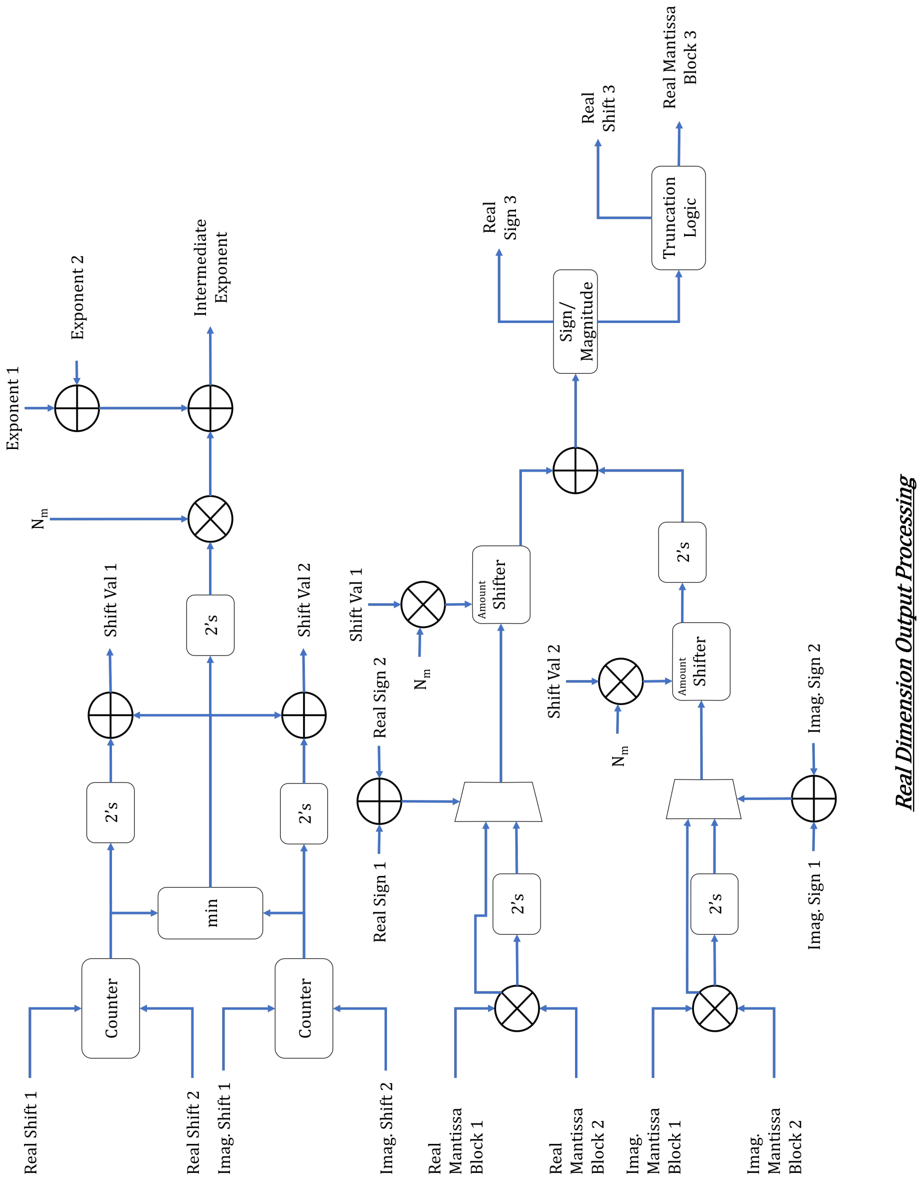


Figure C.3: Complex Block Multiply Block Diagram (i) Real Output

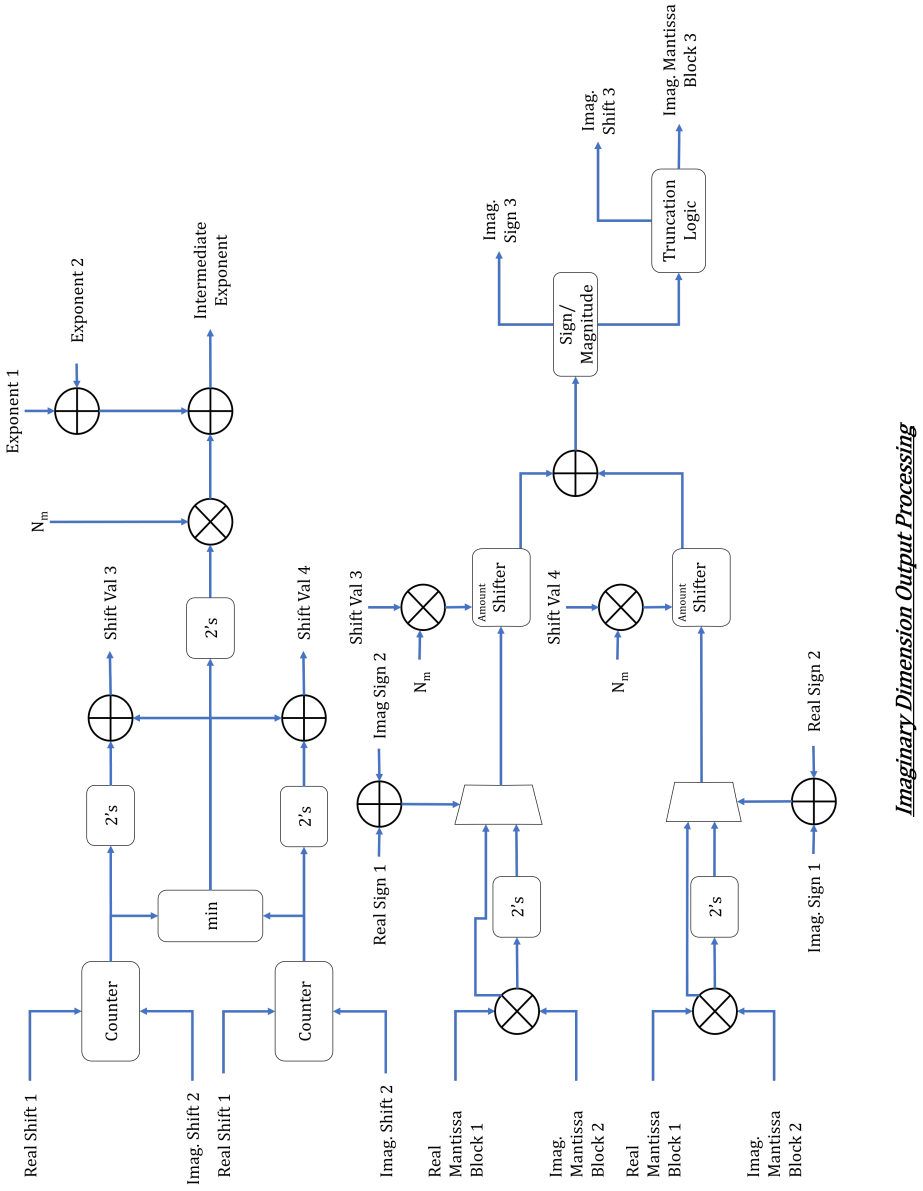


Figure C.4: Complex Block Multiply Block Diagram (ii) Imaginary Output

Bibliography

[1] G. Fettweis and E. Zimmermann, "ICT energy consumption-trends and challenges," in *Proc. Int. Symposium on Wireless Personal Multimedia Communications*, vol. 2, no. 4, 2008, p. 6.

[2] O. Blume, D. Zeller, and U. Barth, "Approaches to energy efficient wireless access networks," in *Int. Symp. on Comm., Control and Sig. Process.*, March 2010, pp. 1–5.

[3] E. Matu, H. Seidel, T. Limberg, P. Robelly, and G. Fettweis, "A GFLOPS Vector-DSP for Broadband Wireless Applications," in *IEEE Custom Integrated Circuits Conference 2006*, Sept 2006, pp. 543–546.

[4] R. H. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 539–550, Apr 1999.

[5] S. Rabii and B. A. Wooley, "A 1.8-V digital-audio sigma-delta modulator in 0.8-um CMOS," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 783–796, Jun 1997.

[6] S. Pavan, R. Schreier, and G. C. Temes, *The Magic of Delta-Sigma Modulation*. Wiley-IEEE Press, 2017, pp. 1–26.

[7] *IEEE Standard for Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers (IEEE) Std., 2008.

[8] N. Cohen and S. Weiss, "Complex Floating Point A Novel Data Word Representation for DSP Processors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 10, pp. 2252–2262, Oct 2012.

[9] R. I. Arun Chhabra, “A Block Floating Point Implementation on the TMS320C54x DSP,” Texas Instruments, Tech. Rep., 1999.

[10] Y. F. Choo, B. L. Evans, and A. Gatherer, “Complex block floating-point format with box encoding for wordlength reduction in communication systems,” in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct 2017, pp. 1023–1028.

[11] Y. F. Choo, “CBFP-Exponent-Box-Encoding,” May 2018. [Online]. Available: https://github.com/yeongfoongc/CBFP-Exponent-Box-Encoding