THE EFFECTS OF FINITE BIT PRECISION FOR A VLSI IMPLEMENTATION OF THE CONSTANT MODULUS ALGORITHM

L. R. Litwin, Jr.*, T. J. Endres†, S. N. Hulyalkar†, and M. D. Zoltowski*

Purdue University* and Sarnoff Digital Communications†

ABSTRACT

One of the most popular blind equalization techniques is the Constant Modulus Algorithm (CMA), and it has gained popularity in the literature and in practice because of its LMS-like complexity and its robustness to non-ideal, but practical, conditions. Although CMA has been well-studied in the literature, these analyses have typically implemented the algorithm using "infinite" precision arithmetic. The motivation for this paper is a VLSI implementation of a high data rate, fractionally spaced, linear forward equalizer whose taps are adjusted using CMA. In this paper we examine how implementing CMA using finite bit precisions affects the algorithm's performance.

1. INTRODUCTION

The Constant Modulus Algorithm (CMA) was originally proposed by Godard [1] for QAM signals and independently by Treichler and Agee [2] for constant envelope FM signals. CMA has seen recent theoretical growth (summarized in [3]) and has also been successfully deployed [4]. Although CMA has been deeply studied in the literature, virtually all of these analyses have been made under the assumption that the algorithm would be implemented using "infinite" precision. We will use the term infinite precision throughout this paper to refer to two cases: 1) when mathematics are performed under the assumption that each number has infinite precision, and 2) when simulations are run using numbers represented with the full precision of the computer. We are interested in the finite precision implementation of CMA and our motivation is a VLSI hardware implementation of CMA using fixed point arithmetic.

Section 2 is a brief primer to CMA and its tap update equation. Section 3 discusses the fixed point arithmetic that is typically used for high-speed hardware designs that involve mathematics. Section 4 describes the assumptions and settings used for the finite precision simulations of CMA. Section 5 presents the simulation results. Section 6 provides concluding remarks.

2. THE CONSTANT MODULUS ALGORITHM

CMA is a gradient descent technique that is used to minimize the cost function described by the Constant Modulus (CM) criterion. This cost function can be written as

$$J_{CM} = E\{(\gamma - |y_n|^2)^2\} \tag{1}$$

where γ is a positive constant known as the Godard radius and y_n represents the equalizer output at baud instance n.

The CMA tap update equation performs a stochastic gradient descent of J_{CM} and it is written as

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mu \mathbf{r}_n^* \underbrace{y_n(\gamma - |y_n|^2)}_{CMA\ error\ term} \tag{2}$$

where μ is a small, positive constant called the stepsize, $\mathbf{f}_n$ is the length-M vector of equalizer taps at time n, and $\mathbf{r}_n = [\, r_n \;\; r_{n-1} \;\; r_{n-2} \ldots r_{n-M+1} \,]^T$ is the regressor vector formed from the M most recent equalizer input samples. Complex conjugation is denoted by the asterisk.

The successful use of CMA in practical receiver implementations has been a motivation for researching reduced complexity versions of CMA. A recent advance in the area of reduced complexity is signed-error CMA [5] which replaces the usual CMA error term with its signum (+/-1). An extension of signed-error CMA is dithered signed-error CMA [6] which introduces a dithering term to the CMA error term prior to the signum operation in order to preserve CMA's robustness properties.

Development of reduced complexity versions of CMA is important for high-speed VLSI applications because reduction in the complexity of the algorithm leads to hardware designs with higher operating speeds

*Department Of Electrical Engineering, Purdue University, West Lafayette, IN 47906, litwin(mikedz)@ecn.purdue.edu

†Sarnoff Digital Communications, Suite 100, 6 Penns Trail, Newtown, PA 18940, endres(samirh)@sdcomm.com

and/or lower gate counts. Hence, since (2) requires a full complex multiply for each equalizer tap, the implementation of (2) to process high data rate signals is often a computational bottleneck and the allocation of bit widths to the components of the CMA update equation is of foremost practical interest. This paper examines the effects of finite bit precision on the performance of CMA by assigning finite bit widths to the components of (2).

3. FIXED POINT ARITHMETIC

This section describes the fixed point representation of numbers and it also lists the major effects of using finite precision arithmetic.

3.1. Representation of Fixed Point Numbers

Although numbers in a VLSI design can be represented in floating point format (mantissa and exponent), fixed point arithmetic is the typical format for high speed designs. In particular, this paper focuses on the effects of implementing CMA using two's complement fixed point arithmetic.

In the two's complement fixed point format, a $B-$bit number's most significant bit (MSB) represents the sign of the number, and the lower $B-1$ bits represent the magnitude. Thus, a $B-$bit number can represent numbers from $-(2^{B-1})$ to $(2^{B-1}-1)$. In two's complement arithmetic, the negative of a binary number is formed by inverting each bit of the number and adding a 1 to the least significant bit (LSB).

3.2. Effects Of Finite Precision Arithmetic

Finite precision arithmetic can have severe effects on the algorithm's performance. In particular, there are three major types of errors that can occur due to finite precision arithmetic [7].

1. Data And Coefficient Quantization Errors

   When representing the data or the filter coefficients with B bits, the total number of possible values that the data or coefficients can take on is 2^B. Hence, the data or the coefficients are quantized to one of the 2^B levels, which introduces quantization noise resulting in a decrease in the effective SNR.

2. Truncation/Rounding Errors

   Truncation and rounding errors occur when a number is converted from a given precision to a lower precision. For example, when multiplying the data and the error term in (2), the bit width of the result increases (multiplying two numbers of bit widths a and b results in a number of bit width $a+b$). In order to maintain a reasonable complexity for the multipliers/adders, this result is truncated and errors are introduced due to the reduced precision.

3. Overflow/Underflow Errors

   Overflow (underflow) errors occur when the result of a calculation is too large (small) to be represented with the allocated bit width. When an arithmetic calculation results in an overflow (underflow), the result is typically clipped [7] to the maximum (minimum) value possible for that number of bits. This clipping operation induces nonlinear distortion in the signal.

The presence of these effects stresses the importance of assigning the appropriate number of bits to the components of each mathematical operation. The simulation results presented in this paper show how the manifestation of the above errors affects the performance of a finite precision VLSI implementation of CMA.

4. ASSUMPTIONS/PARAMETER SELECTIONS

This section describes the assumptions and parameter selections that were used for the simulation results presented in section 5.

4.1. Assumptions

All data is fractionally sampled with a spacing of $T/2$ where T is the symbol period. The channel models are derived from experimentally acquired microwave data and they are available at the SPIB database[1]. Specifically, channels 1 and 2 are used, and they are 300 and 230 samples long, respectively. The source sequence for each simulation consists of 150,000 $T/2$-spaced samples. The 75,000 symbols are taken from an equally probable 64-QAM alphabet. The mean-squared error (MSE) value that is shown in all of the figures is computed by averaging the instantaneous MSE over the last 25,000 symbols.

In order to simulate the fixed point arithmetic used in hardware, the result of each operation is rounded to an integer value and then clipped to make sure that the result is within the range of possible values for the number of assigned bits. The entire computation of the tap update term is performed with a precision of 32

[1]The Rice University Signal Processing Information Base (SPIB) channel database resides at http://spib.rice.edu/spib/microwave.html

bits. This high precision is used to isolate the effects of using reduced precisions for just the data and the taps.

We use B_{data} to denote the number of bits used to represent the data, and B_{taps} to denote the tap bit precision used when multiplying the data in the computation of (2). For the results shown in Figures 1 and 2, the taps are stored at 32-bit (full) precision, but only the upper B_{taps} bits are used to multiply the data. In Figure 3, the taps are stored at reduced precision, and only B_{taps} (where $B_{taps} < 32$) bits are used for both the storage of the tap values, and for the computation of (2).

4.2. Parameter Selections

1. A 16-tap fractionally-spaced equalizer (FSE) is used for Channel 1, and a 32-tap FSE is used for Channel 2.

2. A single spike initialization is used for the taps in all simulations. This initialization involves setting the center tap to unity and all other taps to zero. For the 16-tap equalizer, the center tap is tap position 8, and for the 32-tap equalizer, the center tap is tap position 16.

3. The value used for the step-size is $\mu = 2^{-22}$. This value is specifically chosen to be a power of two because it can be implemented in hardware as a simple right-shift by 22 bits as opposed to an actual multiply.

5. SIMULATION RESULTS

The approach taken in performing the simulations was to first hold the data precision at $B_{data} = 20$ bits while varying the tap precision from $B_{taps} = 4$ bits to 20 bits and recording the MSE for each setting. Subsequently, the tap precision was held at 20 bits while the data precision varied from 4 bits to 20 bits. No noise was present for either case. The purpose of this approach was to determine "good" settings to use for further simulations. The results of these simulations are shown in Figure 1. Note that a tap bit precision of 9 bits yields an MSE similar to that for a tap bit precison of 20 bits. Also note that a data bit precision of 6 bits yields an MSE similar to that for a data precision of 20 bits. The effects of quantization appear when using 5 bits for the data, although an aggressive design might perform adequately using this precision. However, 6 bits is preferable. 5 bits appears to be the lower limit, as evidenced by the jump of over 10 dB for the MSE when only 4 bits are used.

Although these simulations were only done for two channels, from the results we can extrapolate that, in general, the taps are more sensitive to finite bit precision effects compared to the data (this is an expected result based on the analysis of finite bit precision effects on FIR filters in general, for example, see [7]). Due to this increased sensitivity to finite bit precision, the taps of the CMA equalizer should be assigned a higher precision than that assigned to the data. In fact, based on the results shown here and on other simulation results, we hypothesize that a good rule of thumb for the tap bit precision is "data bits plus three," i.e. pick an acceptable value for B_{data} and assign $B_{taps} = B_{data} + 3$ bits for the taps. Based on this rule of thumb, we suggest that $B_{data} = 6$ and $B_{taps} = 9$ would be good precisions to use for 64-QAM and this class of channels, and for a more aggressive design, $B_{data} = 5$ deserves study.

In order to further test this hypothesis, we selected the more aggressive case of $B_{data} = 5$ and $B_{taps} = 9$ and ran simulations with an SNR level of 24 dB and for no noise. The results are presented in Figure 2. The top plot fixes $B_{data} = 5$ and varies the tap precision, while the bottom plot fixes $B_{taps} = 9$ and varies the data precision. Note that for our proposed precisions (and all precisions above that), the noise has the effect of "lifting" the curve. However, for the points below our proposed precisions, the effects of noise do not cause a significant change in performance. This is because the quantization effects dominate at these precisions. From these results, we can state that, for this class of channels, the values of $B_{data} = 5$ and $B_{taps} = 9$ are adequate precisions for 64-QAM, and one might want to use $B_{data} = 6$ to be conservative.

For the results presented above, the taps are stored at a precision of 32 bits, and only the upper B_{taps} are used to multiply the data. Further simulations were run to examine the effects of storing the taps at reduced precisions. The MSE trajectories are shown in Figure 3. In the case of reduced precision, the stated value for B_{taps} is the bit precision used for both storing the taps and used in (2). Note that when the taps are stored at the same reduced precision used to calculate (2), about twice the bit width is needed to achieve similar performance as when the taps are stored at higher precision. These results emphasize the importance of storing the taps at a high precision, even though only a lower precision is needed for the multiplication of the data.

6. CONCLUSIONS

We have shown how a fixed point VLSI implementation of CMA will suffer a performance loss due to the effects of finite bit precisions. Based on an admittedly limited data set, our results suggest that $B_{data} = 6$

bits and $B_{taps} = 9$ bits are good settings to use for 64-QAM, and $B_{data} = 5$ bits should be looked at for an aggressive design. Future work needs to look at how these settings perform on other channels. The results have also shown that the taps need to be stored at a higher precision than what is adequate for multiplying the data. Future work needs to determine a sufficient bit width for the tap storage. The CMA error term was computed using 32 bits, and more simulations need to be run to determine if a smaller bit width still provides adequate performance.

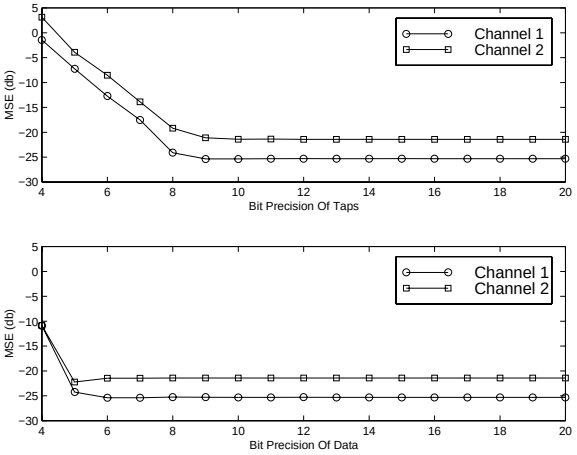

Figure 1: The top plot shows the MSE for $B_{data} = 20$ with varying tap precision and no noise. The bottom plot shows the MSE for $B_{taps} = 20$ with varying data precision and no noise.

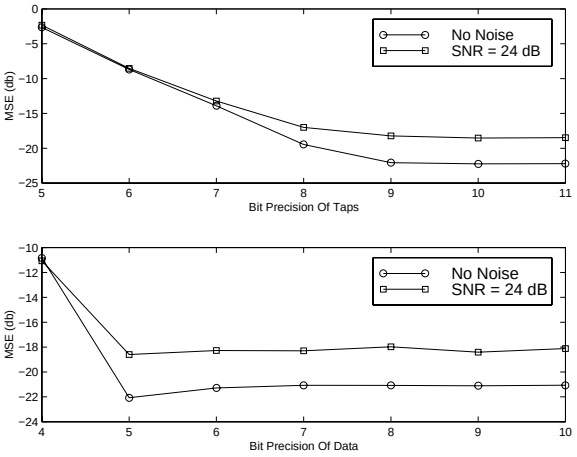

Figure 2: The top plot shows the MSE for $B_{data} = 5$ with varying tap precision. The bottom plot shows the MSE for $B_{taps} = 9$ with varying data precision. Both plots are for Channel 2.

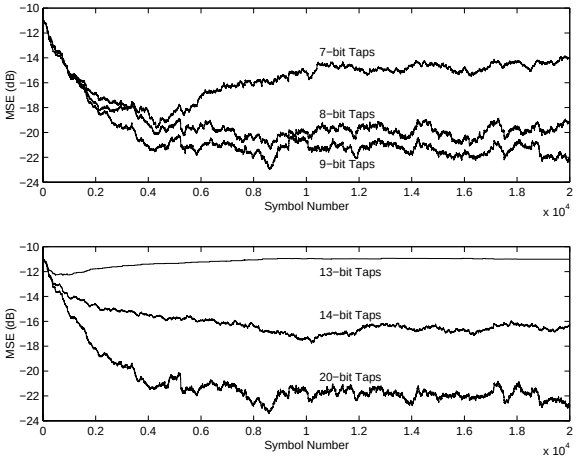

Figure 3: The top plot shows the MSEs that resulted from storing the taps at full (32-bit) precision. The bottom plot shows the MSEs that resulted from storing the taps at reduced ($B_{taps}-$bit) precision. Both plots are for Channel 2 with $B_{data} = 5$ and no noise.

7. REFERENCES

[1] D.N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," IEEE Trans. on Communications, vol. 28, no. 11, pp. 1867-1875, Nov. 1980.

[2] J. R. Treichler, B. G. Agee, "A new approach to multipath correction of constant modulus signals," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-31, no. 2, pp. 459-72, Apr. 1983.

[3] C. R. Johnson, Jr., P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown, R. A. Casas, "Blind Equalization Using The Constant Modulus Criterion: A Review," to appear in Proceedings of the IEEE, Sep. 1998.

[4] J. R. Treichler, M. G. Larimore, J. C. Harp, "Practical blind demodulators for high-order QAM signals," to appear in Proceedings of the IEEE, Sep. 1998.

[5] M. Ghosh, "A Sign-Error Algorithm For Blind Equalization Of Real Signals," International Conf. on Acoustics, Speech and Signal Proc., Seattle, WA, pp. 3365-68, May 12-15, 1998.

[6] P. Schniter, C. R. Johnson, Jr., "The Dithered Signed-Error Constant Modulus Algorithm," to appear in IEEE Transactions on Signal Processing, 1998.

[7] J. G. Proakis, D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, New Jersey: Prentice Hall, 1996.