A Methodology for Exploring Finite-Precision Effects when Solving Linear Systems of Equations with Least-Squares Techniques in Fixed-Point Hardware

Ramon Uribe and Thomas Cesear
AccelChip Inc., ramon.uribe@accelchip.com and tom.cesear@accelchip.com

Abstract

The optimum least-squares solution of linear systems of equations is a fundamental operation in many state-of-the-art communications systems. These include radar, sonar, beamforming, space-time adaptive processing (STAP), Global Position System (GPS) navigation, and multiple-input, multiple-output (MIMO) algorithms where the least-squares optimality criterion is applied. These applications typically require very large amounts of processing which makes implementations in cost-effective, fixed-point hardware – in Field Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs) - the preferred choice. However, the implementation of the least-squares problem in these fabrics has the inherent challenge of sensitivity to finite-precision effects (aka round-off errors) as incurred with fixed-point arithmetic. This paper presents an effective methodology for the exploration of implementation alternatives of least-squares solution of linear systems of equations in fixed-point hardware.

Introduction

An efficient implementation of a least-squares solution depends on essential characteristics of the vectors and matrices that represent the system of equations when cast in a linear algebra context. These characteristics include the size, symmetry or any other structural characteristic of vectors and matrices. These characteristics, along with system requirements for a real-time application drive the selection of a suitable algorithm for implementation.

Traditionally, the implementation of the least-squares problem has been done with general-purpose DSP processors (DSPs) using floating-point arithmetic. Floating-point arithmetic minimizes round-off error making the implementation of a least-squares solution less sensitive to this type of errors. On the other hand, these implementations tend to be limited in processing speed due to the use of a single floating-point processing unit as illustrated in Figure 1 below.

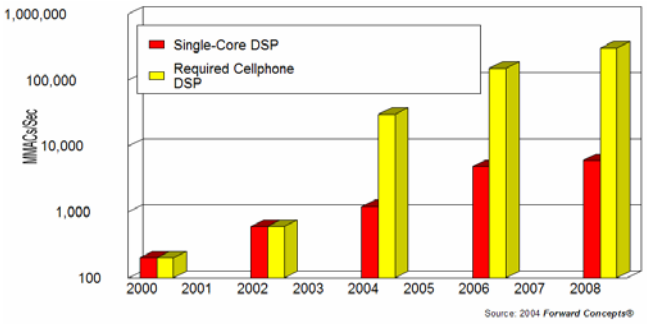

Figure 1: Handset performance requirements vs. DSP processor capability.

The continued success of FPGAs and variations of ASICs in the deployment of high performance DSP algorithms makes them a very appealing implementation fabric. These silicon fabrics, however, are typically tailored for implementations with fixed-point arithmetic. Consequently, the implementation of the least-squares problem in these fabrics has the inherent challenge of sensitivity to round-off errors as incurred with fixed-point arithmetic.

Least-Squares Solution to Linear System of Equations

A linear system of equations can be cast in linear algebra terms as follows:

$$\mathbf{A}x = y$$

where

1) **A** is an *mxn* matrix containing the coefficients of the variables involved in the set of equations,

2) *x* is an *nx1* vector with *n* variables involved in the set of equations,

3) *y* is an *mx1* vector with the equations right hand side values.

Depending on the dimensions of the system, and the rank of the coefficient matrix **A**, the system can have different types of solution (or no solution at all). The specific type of system of equations we will focus on in this paper is the over-determined system of equations. This is the case where the number of equations is larger than the number of unknowns ($m>n$), resulting in a rectangular matrix **A**. This type of system of equations arises in a number of important areas such as radar, sonar, and other sensor array processing applications in general. In these applications, snapshots of sensor data form the rows of the matrix and the number of

columns is determined by the number of sensors in the array.

An over-determined system of equations does not have, in general, an exact solution. The solution to this type of system requires, instead, a solution in the least-squares sense. The following sub-sections outline the use of different matrix factorization techniques to efficiently arrive at the least-squares solution.

QR Factorization

The coefficient matrix **A** can be factor as the product of two matrices

$$\mathbf{A} = \mathbf{QR},$$

where **Q** is an *m*x*m* orthogonal (unitary in complex case) matrix; $\mathbf{QQ}^{-\mathbf{T}} = \mathbf{I}$, and **R** is an *m*x*n* upper-triangular matrix.

This factorization enables solution by back-substitution of the modified system

$$\mathbf{R}x = z,$$

where $z = \mathbf{Q}^{\mathbf{T}}\mathbf{b}$, x can then be computed by back-substitution.

Cholesky Factorization

The Cholesky factorization of a symmetric, positive matrix is given by

$$\mathbf{C} = \mathbf{RR}^{\mathbf{T}},$$

where **R** is an *nxn* upper-triangular matrix called the Cholesky factor of **C**.

To solve the original system of equations $\mathbf{A}x = y$, the covariance matrix of the data in **A** is constructed prior to the solution of the system of equations. In principle, the covariance matrix is defined as

$$\mathbf{C} = \mathbf{AA}^{\mathbf{T}}.$$

In addition, a cross-correlation is computed to define the new right-hand-side of the system of equations. The system of equations can then be solved by finding a solution to the modified system

$$\mathbf{R}x = z,$$

where $z = \mathbf{R}^{\mathbf{T}}$p, and p is the cross-correlation vector. We can then compute x via back-substitution.

SVD

The SVD of an *m*x*n* matrix X is defined as the factorization

$$\mathbf{X} = \mathbf{USV}^{T},$$

where

1) $\mathbf{U} \in R^{mxm}$ is an orthogonal (unitary in the complex case) matrix. The columns of **U** are the left singular vectors of **X**.

2) $\mathbf{V} \in R^{nxn}$ is an orthogonal (unitary in the complex case) matrix. The columns of **V** are the right singular vectors of **X**.

3) $\mathbf{S} = diag(\sigma_1,...,\sigma_p)$ is an *m*x*n* diagonal matrix with $p = min(m, n)$ and $\sigma_1, \sigma_2, .., \sigma_p$ are the singular values of **X**.

To compute a least-squares solution of the system of equations involves creating the Moore-Penrose *pseudo inverse* of **X** given by $\mathbf{X}^{+} = \mathbf{VS}^{+}\mathbf{U}^{T}$, with $\mathbf{S}^{+}$ being diagonal formed with the multiplicative inverses of the non-zero singular values of **X** placed on the diagonal.

Overview of the Methodology

Exploring alternatives early in the design process, while its representation is still at a high level of abstraction, affords the most leverage in terms of impact on the final implementation speed and area cost. Algorithmic and architectural optimization can frequently yield multiple orders-of-magnitude impact on the speed-area solution space of an algorithm. Algorithmic synthesis tools that use a true top-down DSP design methodology enable a collaborative design effort between algorithm developers, system engineers and hardware designers by automating key process steps at different levels of abstraction for an direct implementation in fixed-point arithmetic hardware.

This paper presents an effective methodology for the exploration of implementation alternatives of least-squares solution of linear systems of equations in fixed-point hardware. With the many available choices of algorithms, and the issues related to finite-precision effects in fixed-point arithmetic, the amount of effort required from a design team to arrive at an effective implementation can be formidable. We will describe a fine-grained parameterized model-based library and algorithm synthesis tool that can be used to automate the architecture tradeoff analysis and finite-precision effects allowing the design team to evaluate potential implementation options early and often in the design process. The goal of this methodology is to enable achieving an optimum implementation for a particular application. More specifically, this paper will explore different alternatives for a least-squares solution implementation based on matrix factorization methods. These include triangular-orthogonal (QR) factorization, Cholesky factorization, and singular value decomposition (SVD) techniques. We will demonstrate how this methodology can be effectively used for state-of-the-art communications systems, and we will discuss in detail the architecture, micro-architecture, and finite-precision tradeoff analysis of each of these alternatives.

References

[1] G. Golub, C. Van Loan, *Matrix Computations*, Third Edition, John Hopkins University Press, Baltimore, Maryland, 1996.

[2] J. Proakis et al. *Advanced Digital Signal Processing*, Macmillan Publishing Company, New York, NY, 1992.

[3] G. Strang, K. Borre, *Linear Algebra, Geodesy, and GPS*, Wellesley-Cambridge Press, Wellesley, MA,1997.

[4] D. Rabinkin, W. Song, M. Vai, and H. Nguyen, "Application of Parallel Processors to Read-Time Sensor Array Processing," *Proc. SPIE,* 2001.

[5] R. Walke, R. Smith, and G. Lightbody, "20 GFLOPS QR Processor on a Xilinx Virtex-E FPGA," *Proc. SPIE,* 2000.

[6] D. Martinez, "Application of Parallel Processors to Read-Time Sensor Array Processing," *Proc. IPDPS,* 1999.