EXPLORATION OF LEAST-SQUARES SOLUTIONS OF LINEAR SYSTEMS OF EQUATIONS WITH FIXED-POINT ARITHMETIC HARDWARE

Thomas Cesear and Ramon Uribe
AccelChip, Carlsbad, CA, tom.cesear@accelchip.com / ramon.uribe@accelchip.com

ABSTRACT

One area of focus in Software Defined Radio (SDR) systems is smart antennas. This is due to their ability to provide enhanced communication capacity and minimize interference. The optimum least-squares solution of linear systems of equations is a key operation in state-of-the-art communications systems including smart antenna systems. 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 implementation choice.

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 of the vectors and matrices, symmetry, as well as other structural characteristics. These characteristics, along with system requirements for a real-time application drive the selection of a suitable algorithm for implementation.

1. INTRODUCTION

Traditionally, the implementation of the Least-Squares (LS) solution has been done with general-purpose DSPs using floating-point arithmetic. Floating-point arithmetic minimizes round-off error making the implementation of a LS 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. 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 LS problem in these fabrics has the inherent challenge of sensitivity to round-off errors as incurred with fixed-point arithmetic.

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 LS 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 LS solution implementation based on matrix factorization methods in a beamforming application. These include Cholesky factorization, triangular-orthogonal (QR) factorization, and singular value decomposition (SVD) techniques. We will demonstrate how this methodology can be effectively used for Software Defined Radio (SDR) systems, and we will discuss in detail the architecture, micro-architecture, and finite precision tradeoff analysis of each of these alternatives.

2. BEAMFORMING

Figure 1 shows a basic narrowband beamformer with K sensor elements arranged in a Uniform Linear Array (ULA); this also shows a signal source $\mathbf{s}_\theta(t)$ impinging on the array at an angle of incidence θ. The K beamformer weights $(w_1, w_2, \ldots, w_K)$ are used to linearly combine the array data observation samples $(x_1(n), x_2(n), \ldots, x_K(n))$, and these are set to 'steer' the response of the array for optimum

reception. The output of the beamformer is the scalar output *y(n)*.

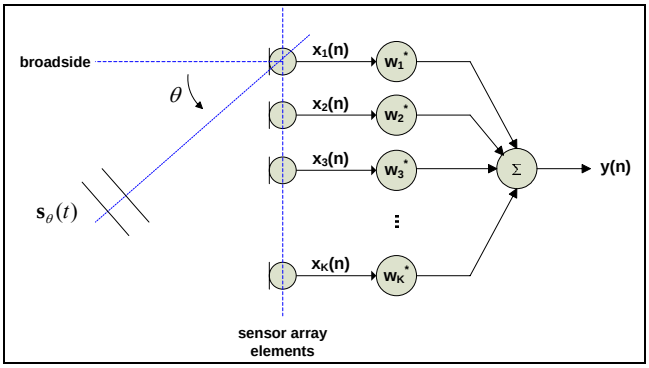

Figure 1 – Narrowband beamformer.

A Generalized Sidelobe Canceller (GSC) is a special beamformer structure that allows the use of unconstrained optimization methods in the design of the optimum beamformer weights [3], [4]. The structure of the GSC is shown in Figure 2.

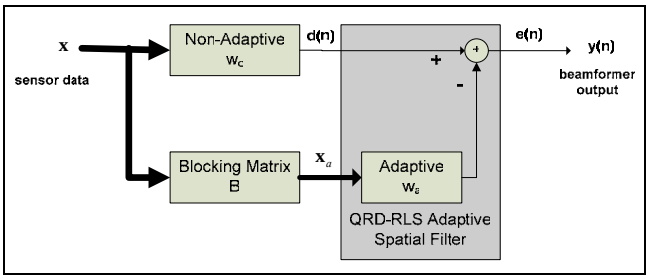

Figure 2 – Generalized Sidelobe Canceller (GSC).

The overall response of the GSC is given by equation (1).

$$y(n) = \mathbf{x}(\mathbf{w}_c - \mathbf{B}\mathbf{w}_a). \quad (1)$$

Here the constant beamformer weights $\mathbf{w}_c$ are designed in a data-independent fashion; the matrix **B** blocks the passing of the input signal of interest in the lower path of the GSC; and the weights $\mathbf{w}_a$ are designed in an optimum manner according to characteristics of the input data.. With the signal of interest removed, the weights $\mathbf{w}_a$ steer their inputs (containing only interfering signals and noise) to generate a signal that is subtracted from the output of the data-independent path. This effectively cancels, in an optimum manner, the interference from the output of the data independent part of the GSC.

When the LS criterion is used, the computation of the optimum beamformer weights $\mathbf{w}_a$ is based on the solution of a system of linear equations known as the deterministic normal equation [1].

$$\mathbf{R}_x\mathbf{w}_a = \mathbf{b}. \quad (2)$$

Here $\mathbf{R}_x$ is the deterministic correlation matrix of the input to the unconstrained section of the GSC, namely $\mathbf{x}_a = \mathbf{x}\mathbf{B}$; and the vector **b** is the cross-correlation of the input $\mathbf{x}_a$ and the ideal response.

The optimum beamformer weights $\mathbf{w}_a$ can then be obtained via inversion of the correlation matrix $\mathbf{R}_x$. From a numerical stability point of view, it is well established that the best approach to matrix inversion is not to do it explicitly whenever possible 0. It is better instead to work with the system of linear equations represented by (2) and then solve this system using an adequate solution technique. A number of effective techniques exist for solving the deterministic normal equation in (2), this paper will focus on three LS solution techniques: Cholesky factorization, QR factorization, and SVD. These techniques are outlined in Section 3. Traditionally, implementations of the solution to (2) have been done with general purpose DSPs and floating-point arithmetic. This type of implementation is less sensitive to round-off errors (finite-precision effects). A key disadvantage of these implementations, however, is the limited processing power they afford due to the small number of floating-point processing units commonly available per device. A very appealing alternative for implementation is to use FPGAs or ASICs which can offer large amounts of parallelism hundreds of computational units per device. One complication with these silicon fabrics is that they are typically tailored for fixed-point arithmetic, and implementation in fixed-point arithmetic is inherently challenging because of sensitivity to finite-precision effects.

3. LEAST-SQUARES SOLUTION TO LINEAR SYSTEM OF EQUATIONS

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

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \quad (3)$$

where:

1) **A** is an *m*x*n* data matrix containing the coefficients of the variables involved in the set of equations,

2) **x** is an *n*x1 vector with *n* variables involved in the set of equations,

3) **y** is an *m*x1 observation vector with the equations right hand side values.

Depending on the dimensions of the system, and the rank of the data 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, usually, an exact solution. The solution to this type of system requires instead some error criterion to judge the optimality of such solution e.g., the minimization of the 2-norm of the error as in a LS solution. The following sub-sections outline the LS solution to an over-determined system of equations and the use of different matrix factorization techniques to find this solution.

3.1. Least-Squares Solution

The LS solution of an over-determined system of equations as defined in (3) is the vector **x** that minimizes the 2-norm of the error. This can be expressed as follows for the case of real-valued matrices and vectors [5]

$$\min_{x \in R^n} \left\| \mathbf{A}x - y \right\|_2 . \tag{4}$$

Differentiating this error measure with respect to the vector **x** results in the symmetric system of equations known as the normal equations.

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y}. \tag{5}$$

When the data matrix **A** has full column rank (i.e., it has linearly-independent columns), the LS solution of the normal equations is unique; multiple techniques to solve equation (5) are available. On the other hand, if **A** is rank deficient, the LS solution is not unique and more specialized techniques are required to find the optimum LS solution.

3.2. LS Solution with Cholesky Factorization

Cholesky factorization is applicable when the data matrix **A** is full rank. To solve the normal equations using Cholesky factorization, the covariance of the data matrix **A** is used in conjunction with the cross-correlation of the matrix **A** and the observation vector. The covariance matrix of **A** is defined as

$$\mathbf{C} = \mathbf{A}^T\mathbf{A}. \tag{6}$$

The cross-correlation is defined as

$$\mathbf{p} = \mathbf{A}^T\mathbf{y}. \tag{7}$$

The normal equations can then be expressed as

$$\mathbf{C}\mathbf{x} = \mathbf{p}. \tag{8}$$

Cholesky factorization can be applied to the covariance matrix **C** when this is positive definite. In such case, this factorization can be expressed as follows

$$\mathbf{C} = \mathbf{R}\mathbf{R}^T, \tag{9}$$

where R is an *n*x*n* upper-triangular matrix called the Cholesky factor of C. Substituting for **C** in (8) we obtain

$$\mathbf{R}^T\mathbf{x} = \mathbf{z}, \tag{10}$$

where

$$\mathbf{R}\mathbf{z} = \mathbf{p}. \tag{11}$$

The LS solution **x** can then be computed via back-substitution in (11) to obtain **z**, and then forward-substitution in (10) using the computed **z.**

3.3. LS Solution with QR Factorization

Triangular-orthogonal factorization – commonly known as QR factorization – is also applicable when the data matrix **A** is full rank. In this technique, the data matrix **A** is factored as the product of two matrices

$$\mathbf{A} = \mathbf{Q}\mathbf{R}, \tag{12}$$

where **Q** is an *m*x*m* orthogonal (unitary in complex case) matrix such that $\mathbf{Q}\mathbf{Q}^{-\mathbf{T}} = \mathbf{I}$, and **R** is an *m*x*n* upper-triangular matrix. The structure of the **R** matrix is of the form

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}, \tag{13}$$

where the $\mathbf{R}_1$ sub-matrix is of dimensions *n*x*n* and **0** is the null matrix of dimensions *(m-n)*x*n.*

Substituting for **A** in the normal equations in (5), we have the equivalent system of equations

$$\mathbf{R}\mathbf{x} = \mathbf{b}, \tag{14}$$

where

$$\mathbf{Q}^T\mathbf{y} = \mathbf{b}. \tag{15}$$

The LS solution can then be found via back-substitution of the reduced system of equations given by

$$\mathbf{R}_1 x = \mathbf{b}_1, \tag{16}$$

where $\mathbf{b}_1$ is the *n*x1 vector containing the first *n* elements of the vector **b**.

3.4. LS Solution with SVD

SVD is applicable to the LS solution even when the data matrix **A** is rank deficient. In general, the SVD of an *m*x*n* matrix **A** is defined as the factorization

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T. \tag{17}$$

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 **A**.

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 **A**.

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

By substituting **A** with its SVD in the normal equations (5) we obtain after simplification

$$\mathbf{S}\mathbf{V}^T\mathbf{x} = \mathbf{U}^T\mathbf{y}. \quad (18)$$

To compute an LS solution of the system of equations involves creating the Moore-Penrose *pseudo inverse* of **A** given by $\mathbf{A}^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T$, with $\mathbf{S}^+$ being a diagonal matrix formed with the multiplicative inverses of the non-zero singular values of **A** placed on the diagonal. The LS solution is then given by

$$\mathbf{x} = \mathbf{A}^+\mathbf{y}. \quad (19)$$

4. GSC BEAMFORMING EXAMPLE

The GSC beamformer floating-point MATLAB model consists of two parts: 1) a top-level script, and 2) a synthesizeable model of the LS algorithms (Cholesky, QR, and SVD). The top-level script generates the input signals and echoes the results to analyze the performance the beamformer, this includes:

- A ULA array of sensors with 4 unity gain, omni-directional elements operating in a narrowband environment.
- A narrowband input signal of interest impinging at an angle of 0^o; this angle is commonly referred to as broadside.
- A narrowband interfering signal impinging at an angle of 10^o and with the same amplitude as the signal of interest. This results in a signal-to-interference ratio of 0 dB.
- Uncorrelated white noise to model receiver noise at a level of -20 dB relative to the signal of interest.

The top-level script performs the data-independent steering of the input sensor data vector as shown in Figure 2. It also applies a blocking matrix **B** to the input to generate the interference-and-noise-only data vector $\mathbf{x}_a$. This script also invokes the various LS algorithms in a streaming fashion to perform the adaptation of the spatial filter for optimum interference cancellation.

The second part of the GSC beamformer MATLAB model are the various synthesizable LS algorithm functions (Cholesky, QR, and SVD) which perform optimum cancellation of the interferer signal.

Figure 3 shows the beampatterns of the GSC. The top plot shows the beampattern of the data-independent portion of the GSC. This shows that the interferer signal impinging at 10^o suffers an attenuation of approximately 2dB relative to that of the desired signal at 0^o; this small attenuation is the cause of the distortion in the received signal from the broadside. The middle plot shows the overall GSC beampattern. The improvement in the cancellation of the interfering signal can be seen with the larger attenuation at 10^o. This is what accounts for the cancellation of the interferer signal obtained at the output of the GSC. The bottom plot is a zoomed view of the overall GSC beampattern to highlight the attenuation around 10^o.

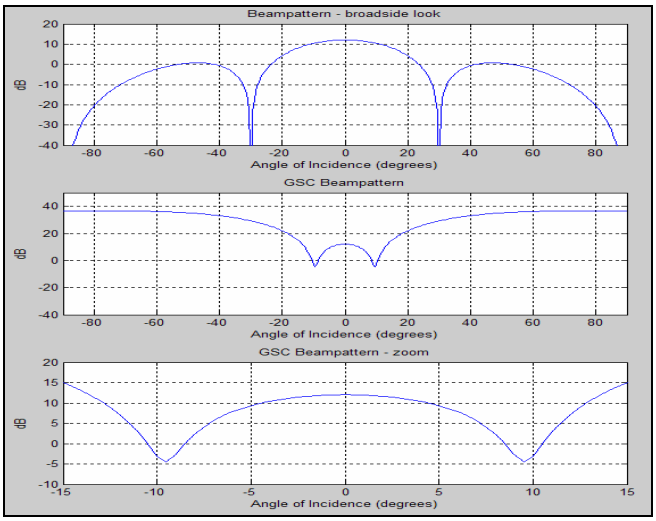

Figure 3 - GSC beam patterns using the QRD-RLS technique.

5. GENERATION OF THE FIXED-POINT MODEL

The starting point in our methodology for obtaining a hardware implementation is the original, golden reference floating-point MATLAB model of the GSC. The next step is to define a fully parameterized fixed-point MATLAB model. This model is directly coupled to the original floating-point MATLAB model to maintain lockstep with this golden reference as we move closer to the actual hardware implementation. There are three critical aspects for efficiency in this step:

1) The ability to intuitively associate fixed-point parameters with variables in the floating-point MATLAB algorithm description. This defines the numerical precision for variables and the operations performed on these in the algorithm.

2) The ability to quickly evaluate the finite-precision effects on the overall performance of the algorithm.

3) Automatic testbench generation to ensure identical functionality between the golden reference floating-point and fixed-point MATLAB models.

It is important to note that the process of defining a fully parameterized fixed-point MATLAB model is typically an iterative process. Iterations in this process aim at minimizing the word-lengths associated with variables and operations in the algorithm to minimize eventual hardware implementation costs. At the same time, the parameterization must be such that the finite-precision effects of the algorithm are minimized.

In the case of the GSC, the numerical performance of the implicit matrix inversion operation is measured by the attenuation shown in the overall beampattern. With this metric, several iterations were performed to define optimum fixed-point arithmetic parameters for each LS algorithm using the flow graph shown in Figure 4. This flow graph is annotated on the right with the key capabilities of the AccelChip DSP Synthesis tool which enable the efficient execution of this step in the methodology.

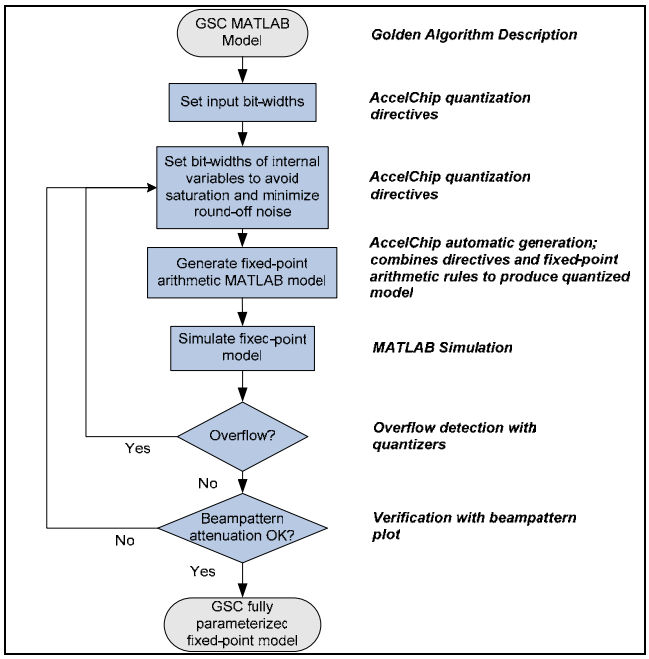

Figure 4 – Fixed-point model definition.

Several input word-lengths were exercised with the intermediate variables sized accordingly to avoid overflows. The effect on the attenuation in the beampattern of the GSC is shown in Figures 5, 6 and 7 for each LS algorithm.

Figures 5, 6 and 7 were used to select the word-lengths for hardware implementation of the various LS algorithms. For the Cholesky factorization technique Figure 5 was used to select a 16-bit implementation, for the QRD-RLS technique Figure 6 was used to select a 16-bit implementation, for the SVD technique Figure 7 was used to select a 13-bit implementation.

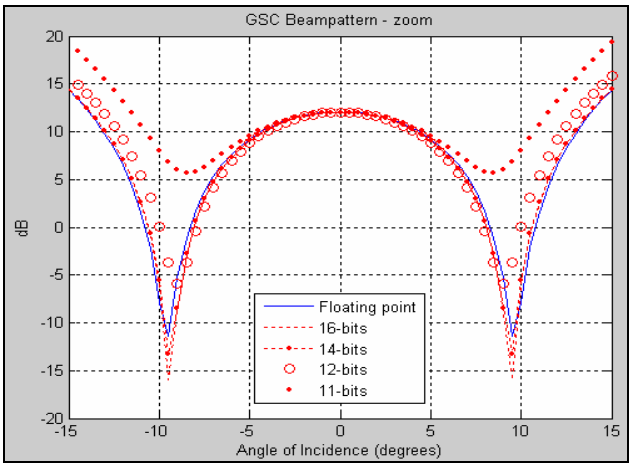

Figure 5 – Finite-precision effects of the Cholesky factorization technique on the GSC Beampattern.

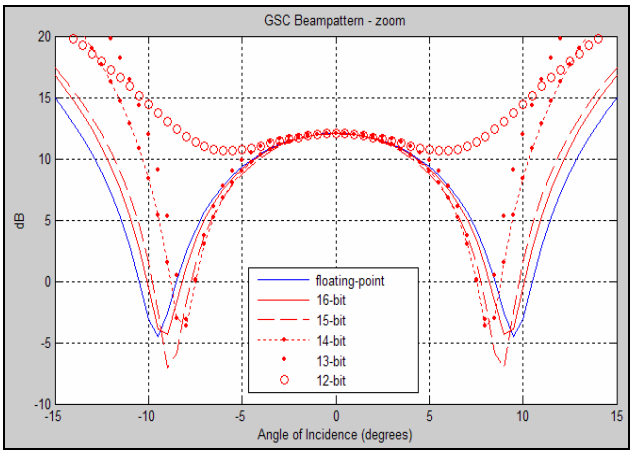

Figure 6 – Finite-precision effects of the QRD-RLS technique on the GSC Beampattern.

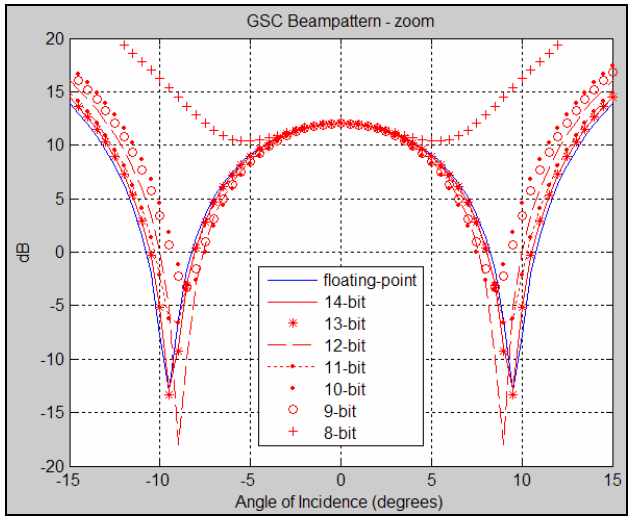

Figure 7 – Finite-precision effects of the SVD technique on the GSC Beampattern.

With this, the end result of this step in our methodology is a fully parameterized, fixed-point MATLAB model of the GSC for each LS algorithm. Equally important, these models have all the information for sizing of signals and arithmetic operations necessary for the generation of a bit-accurate hardware implementation.

6. HARDWARE IMPLEMENTATION

The final step in the methodology is to efficiently generate the hardware implementation. There are two critical aspects to achieve efficiency in this step:

1) The ability to quickly evaluate the impact of hardware resource utilization (e.g., multipliers, pipeline stages, etc.) throughout the algorithm. This ultimately allows one to optimally tailor the hardware architecture of the implementation to meet area/speed requirements.
2) The ability to automatically generate an implementation that is bit-accurate against the fixed-point model of the DSP algorithm. With this, the hardware implementation unambiguously satisfies the numerical precision requirements.

As in the case of defining the fixed-point arithmetic parameters, generation of a suitable hardware implementation is done iteratively. The iterations in this step are aimed at finding the optimum balance of resource utilization and speed of operation to meet the overall system area/speed requirements. The process of generating the hardware implementation using the AccelChip methodology is summarized in the flow graph in Figure 8. This flow graph is annotated on the right with the capabilities of the AccelChip DSP Synthesis tool which enable the efficient execution of this step.

Implementation results for each LS algorithm are shown in Table 1. These results were obtained using Xilinx ISE, targeting a Virtex-4 XC4VSX55 device with an overall goal of maximum speed of operation and minimum use of hardware multipliers.

7. CONCLUSIONS

This paper presented an efficient methodology for the exploration of implementation alternatives of LS solution of linear systems of equations in fixed-point hardware. There are three essential steps in this methodology. First, the capture of the DSP algorithm in a floating-point MATLAB model. Second, definition of fixed-point parameters directly coupled to the floating-point MATLAB algorithm description. Finally, automated generation of a hardware implementation that matches the fixed-point model and meets area/speed requirements.

The foundation for efficiency in the execution of this methodology is the use of the AccelChip DSP Synthesis tool to enable a high level of automation. This was demonstrated via the implementation of a GSC beamformer in an FPGA fabric. The results show the effectiveness of the methodology when used in the implementation of challenging SDR algorithms in fixed-point arithmetic hardware.

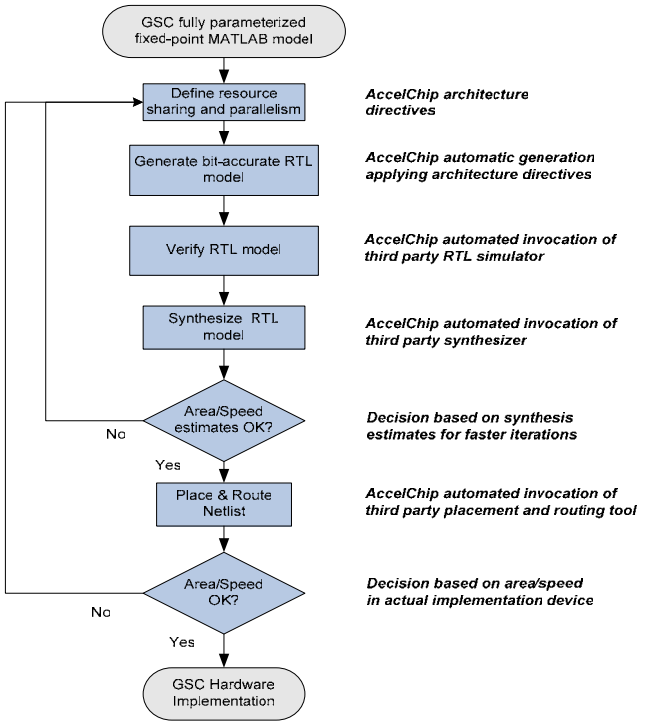

Figure 8 – Hardware implementation generation.

Table 1 - Implementation results.

	Cholesky	QR	SVD
Occupied Slices	1011 (4%)	3076 (12%)	8926 (36%)
DSP48s	37	1	129
Sustainable data rate	0.07 Msps	1.7 Msps	0.04 Msps

8. REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[2] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 2002.

[3] D.H. Johnson and D.E. Dudgeon, *Array Signal Processing Concepts and Techniques*, Prentice-Hall, Upper Saddle River, New Jersey, 1993.

[4] B.D. VanVeen and K. Buckley, "Beamforming: A Versatile Approach to Spatial Filtering," *IEEE ASSP Magazine*, pp. 4-24, April 1988.

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

[6] D. Rabinkin, W. Song, M. Vai, and H. Nguyen, "Adaptive Array Beamforming with Fixed-Point Arithmetic Matrix Inversion using Givens Rotations," *Proc. SPIE,* 2001.