Revisiting the Adjoint Matrix for FPGA Calculating the Triangular Matrix Inversion

Di Yan, Wei-Xing Wang, Lei Zuo, *Member, IEEE*, and Xiao-Wei Zhang

***Abstract*—Due to the robustness, the matrix inversion methods based on matrix factorization are often adopted in engineering while the triangular matrix inversion is one key part of those methods. This brief reviews the adjoint matrix and presents a novel scheme for field-programmable gate arrays (FPGAs) calculating the triangular matrix inversion. Employing the more characteristics of both the triangular matrix and its inversion, the proposed diagonal-wise algorithm for the triangular matrix inversion has the high parallelism and extensibility in the hardware implementation and is suitable for the different matrix triangular factorization (QR, LDL, Cholesky and LU). Meanwhile, the recursive diagonal-wise algorithm is designed for the large scale triangular matrices. Compared with the traditional row-/column-wise methods, our algorithm has a good performance at the low computation load. Finally, the experiments are conducted on one Xilinx Virtex-7 FPGA to illustrate the performances of the four different methods.**

***Index Terms*—Adjoint matrix, field-programmable gate array (FPGA), matrix inversion, triangular matrix.**

I. Introduction

THE MATRIX inversion is one basic mathematical tool and plays a key role in many applications, such as: auto-control systems [1], signal processing [2] and communications [3], to name a few. It is well known that the high dimensional matrix inversion has a big computation load. Thus, both the researchers and engineers try to avoid the annoying problem and look for some new algorithms without the matrix inversion (or reducing the matrix dimension) [4], [5] in their fields. However, we are still confronted with the real-time calculation of the high dimensional matrix inversion in some scenarios. For example, engineers have to achieve some algorithms (adaptive digital beam former (ADBF) [6] or model predictive control [7], etc.) containing the matrix inversion on the hardware platform. In the past years, the digital integrated circuits have made a remarkable progress, which provides a good foundation for solving this problem efficiently in engineering. Due to the high energy efficiency ratio [8], this brief is focused on the implementation of the triangular matrix inversion on field-programmable array gates (FPGAs).

Compared with the standard definition method, those matrix inversion algorithms based on matrix factorization are numerical stability and have the low computation load. In general, the matrix to be calculated its inversion is often decomposed into the orthogonal or triangular matrices. For one non-Hermitian matrix, it can be decomposed by the LU or QR factorization. The Hermitian matrix can be decomposed by the Cholesky or LDL factorization, which could also be processed by the LU or QR factorization. Then, either back substitution or equation solving is usually used to solve the triangular matrix inversion. The above process is traditionally adopted by engineers [9]–[11]. Hence, the triangular matrix inversion is one important part of those methods. Note that the relationship of the diagonal elements between the triangular matrix and its inversion is used to avoid computing the intermediate results in [12]. As shown in [13], the authors proposed a novel method based on the Neumann series expansion which is just suit for the special scenario. The authors of [14] proposed a simple positive definite symmetric matrix inversion algorithm (SPMI), which is actually a simple fixed form of the block matrix inversion. In fact, the traditional methods are processing the column- or row-vector step by step belonging to either the row- or column-wise algorithms.

For the triangular matrix inversion, this brief reviews its adjoint matrix and we find that the more characteristics of the original matrix and its inversion can be used in the calculation process. The main goal of this brief is solving the triangular matrix inversion and presents the new scheme for FPGA calculating the triangular matrix inversion. To this end, the new diagonal-wise algorithm for the triangular matrix inversion is proposed to reduce the computation load and improve the hardware implementation efficiently in this brief. On the other hand, the proposed algorithm has no close relationship with the kinds of the matrix factorization. Here, it should be pointed out that for the high dimensional matrices, we also give the quasi-diagonal-wise algorithm based on the recursive block matrix inversion to achieve the annoying problem. When we have programmed and achieved the matrix of size $N \times N$ by our algorithm in hardware, there just needs a few modifications for the old programming if the matrix size is changed to $(N+1) \times (N+1)$. In other words, the proposed recursive quasi-diagonal-wise algorithm will be very popular for engineers. In addition, supplementary material provides the

Manuscript received June 28, 2020; revised August 19, 2020 and September 26, 2020; accepted October 25, 2020. Date of publication October 28, 2020; date of current version May 27, 2021. This work was supported by the National Natural Science Foundation of China under Grant 618713071. This brief was recommended by Associate Editor H.-G. Yang. *(Corresponding author: Xiao-Wei Zhang.)*

Di Yan and Wei-Xing Wang are with the School of Information and Communication, Chang'an University, Xi'an 710064, China (e-mail: ydxust@126.com; znn525d@qq.com).

Lei Zuo is with the National Laboratory of Radar Signal Processing, Xidian University, Xi'an 710071, China (e-mail: lzuo@mail.xidian.edu.cn).

Xiao-Wei Zhang is with the School of Communications and Information Engineering, Xian University of Posts and Telecommunications, Xi'an 710121, China (e-mail: xwzhang@stu.xidian.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCSII.2020.3034437.

Digital Object Identifier 10.1109/TCSII.2020.3034437

FPGA implementation of ADBF for one 1-D phased-array radar.

The organization of this brief is as follows. The traditional methods are introduced in Section II. The new algorithm is described in Section III. Section IV shows the experimental results and Section V summarizes this brief.

II. Matrix Factorization and Back Substitution

In this section, we simply review the traditional triangular matrix inversion methods based on matrix factorization. For a non-singular matrix $\mathbf{A} \in \mathcal{C}^{N\times N}$, it can be decomposed by the LU or QR factorization: $\mathbf{A} = \mathbf{LU} = \mathbf{QR}$, where $\mathbf{U}$ and $\mathbf{R}$ are upper triangular; $\mathbf{L}$ and $\mathbf{Q}$ denote the lower triangular matrix and orthogonal. When $\mathbf{A}$ is a non-singular Hermitian matrix, it can also be decomposed in the following form: $\mathbf{A} = \mathbf{G}^H\mathbf{G} = \mathbf{P}^H\mathbf{DP}$, where $\mathbf{G}$ and $\mathbf{P}$ are upper triangular; $\mathbf{D}$ is the diagonal matrix. Then, back substitution is usually adopted to solve the triangular matrix inversion. For an upper triangular matrix $\mathbf{B}$, its inversion $\mathbf{T}$ can be computed by

$$T_{i,j} = \begin{cases} -\left(\sum_{k=1}^{j-1} T_{i,k}b_{k,j}\right)/b_{j,j} & i < j \\ \frac{1}{b_{i,j}} & i = j \\ 0 & i > j \end{cases} \tag{1}$$

where i and $j(1 \le i, j \le N)$ are the coordinate of elements in the matrix. Obviously, equation (1) is easily understood but has the lower efficiency in the hardware implementation. The two main reasons are that the intermediate results of the current procedure are dependent on that of the previous step and that the existing method has the lower parallelism.

It is well known that the diagonal elements of triangular matrices obtained by LDL factorization are equal to 1, which is not applied to other matrix factorization methods. Here $\mathbf{W}$ is assumed to be the triangular matrices obtained by other methods. Then $\mathbf{W}$ can be transformed into one triangular matrix whose diagonal elements are equal to 1 and the transformation matrix is produced by the diagonal elements of $\mathbf{W}$. The above procedure can be expressed as

$$\mathbf{W} = \mathbf{VZ} \tag{2a}$$

$$\mathbf{W}^{-1} = \mathbf{Z}^{-1}\mathbf{V}^{-1} \tag{2b}$$

where $V = diag(\mathbf{W})$, $diag(\bullet)$ creates a matrix with the elements on the main diagonal. $\mathbf{Z}$ can be expressed as $\mathbf{Z} = [Z_1, Z_2, \ldots Z_N]^T$, where $[\bullet]^T$ denotes the transposition operation andZ_i is the i-*th* row vector of $\mathbf{Z}$. If we have obtained $\mathbf{Z}^{-1}$, W_i^{-1} can be computed by $Z_i^{-1}/v_{i,i}$ which is easily processed and programmed in hardware. Thus, the scenario that the diagonal elements of the upper triangular matrix are equal to 1 is considered in this brief. The reasons can be simply concluded as the follows:

1) the diagonal elements do not participate in the intermediate calculation process in order to reduce the rounding-off errors;
2) the assumed scenario is conducive to save the memory resource when the matrix dimension is high;
3) the considered upper triangular matrix can reduce the operation numbers and computation load.

For the proposed algorithm, the calculation process of the upper triangular matrix is similar to that of the lower triangular matrix. Therefore, we will just discuss the upper triangular matrix in the following sections.

III. The Proposed Diagonal-Wise Algorithm

In this section, we initially review the adjoint matrix of the upper triangular matrix and its inversion in the first part. Then the proposed (quasi-) diagonal-wise algorithm for the upper triangular matrix inversion is described in the following parts.

A. Adjoint Matrix and Upper Triangular Matrix Inversion

For an upper triangular matrix $\mathbf{B}$, it can be expressed as

$$\mathbf{B} = \begin{bmatrix} 1 & b_{12} & \ldots & b_{1N} \\ 0 & 1 & \ldots & b_{2N} \\ \vdots & 0 & 1 & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3}$$

where $b_{i,j}(1 \le i, j \le N)$ denotes the non-zero elements of $\mathbf{B}$. Based on the definition of the matrix inversion, the inverse matrix of $\mathbf{B}$ [15] can be written in the following expression

$$\begin{aligned} \mathbf{B}^{-1} &= \frac{\text{adj}(\mathbf{B})}{\det\ (\mathbf{B})} \\ &= \frac{\text{cof}(\mathbf{B})^T}{|\mathbf{B}|} = \frac{1}{|\mathbf{B}|}\begin{bmatrix} B_{1,1} & B_{1,2} & \ldots & B_{1,N} \\ B_{2,1} & B_{2,2} & \ldots & B_{2,N} \\ \vdots & \vdots & \ldots & \vdots \\ B_{N,1} & B_{N,2} & \ldots & B_{N,N} \end{bmatrix}^T \end{aligned} \tag{4}$$

where $\det(\cdot)$, $\text{adj}(\cdot)$ and $\text{cof}(\cdot)$ are the matrix determinant, adjoint matrix and matrix cofactor, respectively. $B_{i,j} = 0$, when i is smaller than j. Thus, equation (4) can be rewritten as

$$\begin{aligned} \mathbf{B}^{-1} &= \begin{bmatrix} B_{1,1} & 0 & \ldots & 0 \\ B_{2,1} & B_{2,2} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ B_{N,1} & B_{N,2} & \ldots & B_{N,N} \end{bmatrix}^T \\ &= \begin{bmatrix} B_{1,1} & B_{2,1} & \ldots & B_{N,1} \\ 0 & B_{2,2} & \ldots & B_{N,2} \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & B_{N,N} \end{bmatrix} \end{aligned} \tag{5}$$

From (5), it is easily seen that the upper triangular matrix inversion has the same structure with itself. Because the matrix determinant of (3) is equal to 1, we can directly acquire the following form:

$$\mathbf{B}^{-1} == \begin{bmatrix} 1 & B_{2,1} & \ldots & B_{N,1} \\ 0 & 1 & \ldots & B_{N,2} \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \tag{6}$$

Based on the forms of (6) and (3), the authors of [12] proposed a good algorithm to avoid computing the intermediate results.

In fact, the following characteristic can also be used in the calculation process of the triangular matrix inversion. For $B_{i+1,i}$ it can be computed by the adjoint matrix definition

$$\begin{aligned} B_{i+1,i} &= \left((-1)^{2i+1}b_{i,(i+1)}\prod_{n=1,n\neq i}^{N} b_{n,n}\right)\Big/\left(\prod_{n=1}^{N} b_{n,n}\right) \\ &= -b_{i,(i+1)} \end{aligned} \tag{7}$$

From (7), it can be observed that $B_{i+1,i}$ is the reverse of $b_{i,(i+1)}$. Now, we can say that the number of non-zero elements in $\mathbf{B}^{-1}$ to be computed is $0.5(N^2\text{-}3N + 2)$, because its

diagonal elements are equal to 1 and $B_{i+1,i}$ is the reverse of $b_{i,(i+1)}$.

Before introducing our new algorithm, we firstly take three examples to illustrate the characteristics of the upper triangular matrix and its inversion. The considered matrix size is 3×3, 4×4 and 5×5, respectively.

When $N = 3$, the non-zero element of $\mathbf{B}$ needed to be calculated is computed by

$$B_{3,1} = b_{1,2}b_{2,3} - b_{1,3} \tag{8}$$

when $N = 4$, the non-zero elements of $\mathbf{B}$ needed to be calculated are computed by

$$B_{3,1} = b_{1,2}b_{2,3} - b_{1,3} \tag{9a}$$
$$B_{4,2} = b_{2,3}b_{3,4} - b_{2,4} \tag{9b}$$
$$B_{4,1} = -\left[b_{1,2}\left(b_{2,3}b_{3,4} - b_{2,4}\right) - \left(b_{1,3}b_{3,4} - b_{1,4}\right)\right] \tag{9c}$$

when $N = 5$, the non-zero elements of B needed to be calculated are computed by

$$B_{3,1} = b_{1,2}b_{2,3} - b_{1,3} \tag{10a}$$
$$B_{4,2} = b_{2,3}b_{3,4} - b_{2,4} \tag{10b}$$
$$B_{5,3} = b_{3,4}b_{4,5} - b_{3,5} \tag{10c}$$
$$B_{4,1} = -\left[b_{1,2}\left(b_{2,3}b_{3,4} - b_{2,4}\right) - \left(b_{1,3}b_{3,4} - b_{1,4}\right)\right] \tag{10d}$$
$$B_{5,2} = -\left[b_{2,3}\left(b_{3,4}b_{4,5} - b_{3,5}\right) - \left(b_{2,4}b_{4,5} - b_{2,5}\right)\right] \tag{10e}$$
$$B_{5,1} = b_{1,2}\left[b_{2,3}\left(b_{3,4}b_{4,5} - b_{3,5}\right) - \left(b_{2,4}b_{4,5} - b_{2,5}\right)\right] - \left[b_{1,3}\left(b_{3,4}b_{4,5} - b_{3,5}\right) - \left(b_{1,4}b_{4,5} - b_{1,5}\right)\right] \tag{10f}$$

For (8), we can say that there is one complex multiplication and one subtraction operation. From (9) and (10), we can also acquire two characteristics of the calculation process in the upper triangular matrix inversion:

1) $B_{i+m,i}$ has the same calculation structure with the different entries of B, where $m$is from 3 to N. In other words, non- zero elements of the triangular matrix inversion based on its adjoint matrix is computed by the different determined location entries.
2) The previous calculation results can also be reused in the next computing process. To illustrate this characteristic, we review the equations (10) and can find that the equations of both (10b) and (10c) reappear in the equations of (10d) and (10e). Meanwhile, the equation (10e) also reappears in the equation (10f). In other words, we can delete the redundant calculation in the computing process of the upper triangular matrix inversion.

Based on the adjoint matrix definition [15], the elements of B actually can be expressed by

$$B_{j,i} = 0, \quad i < j \tag{11a}$$
$$B_{j,i} = 1, \quad i = j \tag{11b}$$
$$B_{j,i} = -b_{i,j}, \quad i = j - 1 \tag{11c}$$
$$B_{j,i} = b_{i,j-1}b_{i+1,j} - b_{i,j}, \quad i = j - 2 \tag{11d}$$
$$B_{j,i} = (-1)^{i+j}\left\{b_{i,i+1}(b_{i+1,j-1}b_{j-1,j+1} - (b_{i,j}b_{i+1,j-1} - b_{i,j}))\right\}, \quad i = j - 3 \tag{11e}$$
$$\vdots$$

Equation (11) is the analytical expression of the adjoint matrix for the upper triangular matrix. In reality, we often process the fixed dimensional matrix in engineering. Thus, we can employ the characteristics described in this part to acquire the inverse of the upper triangular matrix.

TABLE I
THE NUMBERS OF OPERATIONS FOR DIFFERENT METHODS

Method	Equation Solving	Triangular matrix Operation	[12]	Our algorithm
Operations	$\frac{5}{6}n^3$	$\frac{2}{3}n^3$	$\frac{1}{2}n^3$	$\frac{1}{3}n^3$

Finally, we can conclude that if we use these characteristics in the calculation process of the upper triangular matrix inversion, it will save the precious hardware resource and reduce the low computation load. In fact, the main difference between our algorithm and the traditional methods is that our algorithm calculates the elements parallel to the main matrix diagonal orderly whereas the traditional methods computes the row or column vectors step by step. When the matrix dimension is high, it is not easy for engineers to achieve the programming in the hardware implementation. One main reason is that it is too long for the analytical expressions calculating these elements farthest from the main diagonal, for which there is no need to compute them via a big amount of hardware resources. Therefore, we propose the recursive diagonal-wise algorithm for the large scale triangular matrix in the next part.

B. Recursive Diagonal-Wise Algorithm

In the previous part, we have analyzed the characteristics in the calculation process of the upper triangular matrix inversion. If the size of upper triangular matrices is small, we can directly calculate its inversion based on the characteristics. However, when the upper triangular matrix to be processed is a large scale matrix, the above method may not be a good idea. On the other hand, we often have programed the matrix of size $N \times N$, while the other project needs to process the matrix of size $(N+k)\times(N+k)$, where k is a positive integer. Hence, we advise that it com- bine the block matrix inversion method [16] with the above characteristics and consider the form of a block matrix:

$$\mathbf{B} = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{O} & 1 \end{bmatrix} \tag{12}$$

where the size of matrix $\mathbf{X}$ is $(N-1)\times(N-1)$, $Y \in \mathcal{C}^{(N-1)\times 1}$ is a column vector and $\mathbf{O} \in \mathcal{C}^{1\times(N-1)}$is a row zero vector. Its inversion is determined by

$$\mathbf{B}^{-1} = \begin{bmatrix} \mathbf{X}^{-1} & -\mathbf{X}^{-1}\mathbf{Y} \\ \mathbf{O} & 1 \end{bmatrix} \tag{13}$$

From (13), it can be observed that if we have computed $\mathbf{X}^{-1}$, the next is just a matrix multiplied by one vector.

Thus, our algorithm can be described as follows: the first part is employing those characteristics to calculate the upper triangular matrix of small size and equation (13) is then used to process the remaining calculation processes. We take $B_{6\times6}$ for an example to illustrate the process of our algorithm. $B^{-1}_{4\times4}$ is computed by the equation (11) and we then use equation (13) at two times to obtain $B^{-1}_{5\times5}$ and $B^{-1}_{6\times6}$ respectively. Finally, we show the numbers of operations for the different methods in Table I. It can be seen that our algorithm has the lowest operation numbers.

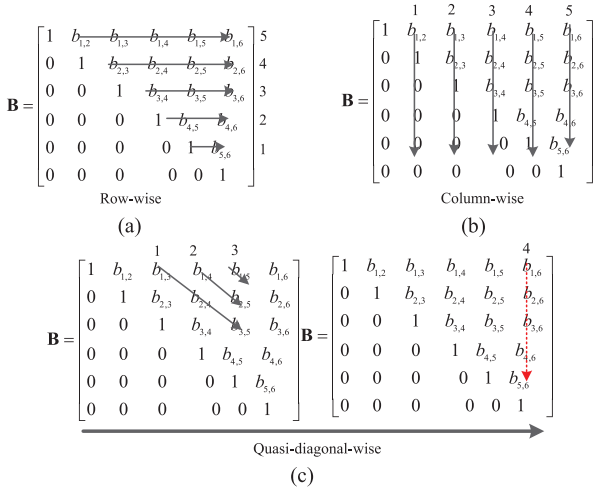

Fig. 1. Sketch maps of three methods for the upper matrix inversion and the matrix size is 6 × 6. (a) The row-wise algorithm. (b) The column-wise algorithm. (c) Our algorithm.

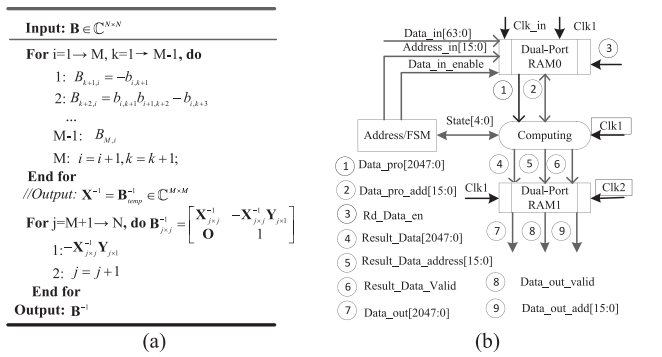

Fig. 2. The hardware implementation. (a) The top-level pseudo code. (b) The diagrammatic for the hardware implementation.

C. Comparison of the Different Methods

In this part, we initially show the sketch maps of three methods (the row-wise, column-wise and our algorithm) for the upper triangular matrix inversion in Fig. 1.

From Fig. 1(a), it can be observed that the flowchart is often adopted by those methods based on the QR or Cholesky factorization in engineering, where there needs many memory resources to save the intermediate results and it is often performed by the systolic array in hardware. For the column-wise algorithm shown in Fig. 1(b), it is also confronted with the same problem. From Fig. 1(c), we can see that the number of non-zero elements to be calculated by our algorithm is smallest among three methods, where we steal the characteristics described in the last part to delete the redundant operations. The annoying problem here is that the write/read addresses of different entries are complex in the different calculation steps. However, the changing addresses can be computed in advance for the different steps.

Finally, we describe the main novelties of our algorithms as follows: the vectors parallel to the main diagonal in the first part of our algorithm are independent of each other, which means that our algorithm can compute the whole of all non-zero elements at one clock period in theory; the simple operation of the second part in our algorithm is one matrix multiplied by a column vector, which can also be performed at one clock period in theory. For the FPGA design, however, there is the trade-off among the area, speed and power. Thus, the selection of the parallelism level in reality is decided by many factors such as: design indexes, chip level and cost, etc.

IV. Hardware Architecture and Experiments

In this section, we first give the top-level description for the hardware implementation of the upper triangular matrix inversion. In addition, the hardware implementation on one Xilinx Virtex-7 (XC7VX690T) is described in the first part of this section. It should be pointed out that the experimental platform in this brief is the same PCB board in [6].

A. Top-level Description and Hardware Implementation

In this part, the top-level description for the hardware implementation of the upper triangular matrix inversion is shown in Fig. 2(a). From Fig. 2(a), it can be observed that there are two for-loops performed in the proposed algorithm. The first loop is that we employ the characteristics to calculate the upper triangular matrix inversion and the last one is that the recursive part for the triangular matrix inversion is adopted to achieve the remaining computation. For the first for-loop, $B_{i+k,i}$ is the cofactor of $\mathbf{B}$ with the (i+k, i)-position. Meanwhile, j, the intermediate variable quantity, denotes the size of matrix $\mathbf{X}_{j\times j}$.

Due to the space, Fig. 2(b) shows the diagrammatic for the hardware implementation on one Xilinx Virtex-7 FPGA where we present some key signals among the different modules. From Fig. 2(b), we can see that there are mainly four parts in the proposed algorithm. The dual-port RAM0 is used to cache the data to be processed and the Computing module is used to achieve the upper triangular matrix inversion mainly composed of the complex multiplication and addition operations, while the address/finite state machine (FSM) module generates the different states for the different calculation procedures and control the whole circuit system. Meanwhile, the dual-port RAM1 is used to cache the calculation result and transfer the result to the other chip. Finally, we list some differences among our algorithm and the traditional methods in the following.

1) Our algorithm has the lowest computation load, because we can delete the redundant operations in the calculation process. However, the traditional methods do not share this feature.
2) The most complex part is the address part in our algorithm. The processed matrix size is often fixed in engineering, so its address map is predetermined and can also be saved in one ROM or computed in each step.
3) One big advantage of our proposed algorithm is that the two calculation procedures have the high level parallelism.

B. Experimental Results

Here the four matrix inversion methods are selected and programmed to illustrate those performances, which are composed of the back substitution, SPMI, the proposed method in [6] and our algorithm. For the sake of fairness, the numerical calculation for four methods is in the single precision float-point arithmetic (IEEE Std. 754-1985). In addition, the descriptions about the experimental PCB board can be referred to [6].

TABLE II
UTILIZATION-POST IMPLEMENTATION OF THE FIRST EXPERIMENT

Resource	Back substitution	SPMI	[6]	Our Algorithm
LUT	29130	16856	15867	15415
FF	33380	26276	21764	18048
BRAM	14	15	13	10
DSP48	184	160	132	104
Clock cycle	7701	6547	5094	3621
f_{max}, (MHz)	250	250	250	300

TABLE III
UTILIZATION-POST IMPLEMENTATION OF THE SECOND EXPERIMENT

Resource	Back substitution	SPMI	[6]	Our Algorithm
LUT	274780	234856	21498	210781
FF	257076	240896	22659	214792
BRAM	321	278	235	206
DSP48	1768	1429	1296	1168
Clock cycle	123260	105986	99310	88796
f_{max}, (MHz)	200	200	200	250

The first experiment is conducted when the size of matrix is 8×8 and the utilized resources of the four methods after the post implementation are listed in Table II and there is no recursive step in our algorithm. From Table II, we can see that our algorithm utilizes the least resources but can work at the highest frequency.

In fact, we often have to process the high dimensional matrix in engineering. Thus, we will achieve the big size of matrix inversion in the following experiment. Here the size of matrix is 32×32 and the first for-loop achieves the matrix inversion of size 10×10 in our algorithm. Then, the second experimental results are shown in Table III.

From Table III, we can acquire the same conclusion from this experiment with that from the first one. From the two above experimental results, it can be observed that our algorithm can work at the highest frequency. One main reason is that our algorithm steals the more characteristics of both the upper triangular matrix and its inversion in its calculation procedure. Thus, our algorithm consumes the least resources so that it can work at the highest frequency.

The last experiment is conducted to illustrate the robustness of our algorithm. It is known that numerical stability is very important in engineering. Due to the propagation of rounding-off errors [16], it is very difficult for us to analyze those methods containing a lot of mathematical operations. Hence, we use the matrix 2-norm errors,$c = \|\mathbf{I} - \mathbf{B}\mathbf{B}^{-1}\|_2$, to illustrate the performances of the four methods, where $\| \|_2$ and I denotes the matrix 2-norm and unit matrix, respectively. In this experiment, we use the numbers generated by the Randn function in MATLAB as the test data and the experimental results of 1000 upper triangular matrices tested by the FPGA are shown in Fig. 3. From Fig. 3, it can be observed that our algorithm has the lowest amplitudes of matrix 2-norm errors.

V. CONCLUSION

In this brief, the novel scheme steals the more characteristics of both the triangular matrix and its inversion to reduce the computation load. Meanwhile, the proposed algorithm has been used to solve the ADBF for one 1-D phased-array radar. Please see supplementary material.

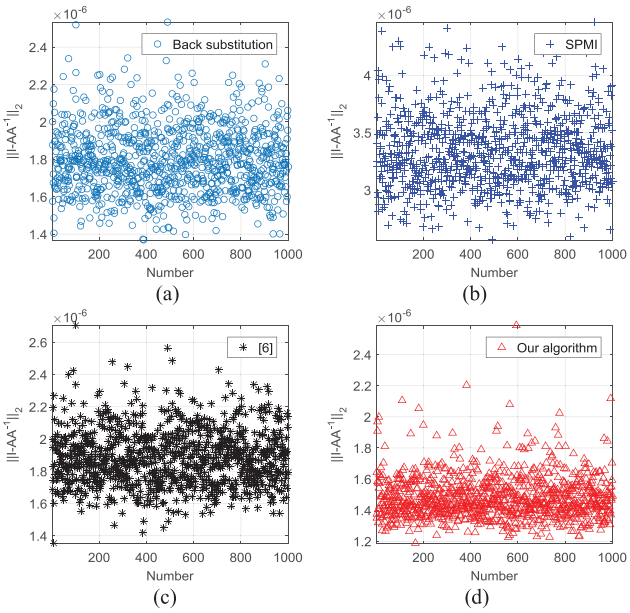

Fig. 3. Matrix 2-norm errors of the four methods. (a) Back substitution. (b) SPMI. (c) The proposed method in [6]. (d) Our algorithm.

REFERENCES

[1] B. Kulcsar and M. Verhaegen, "Robust inversion based on fault estimation for discrete-time LPV systems," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1581–1586, Jun. 2012.
[2] Z.-Y. Huang and P.-Y. Tsai, "Efficient implementation of QR decomposition for gigabit MIMO-OFDM systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2531–2542, Oct. 2011.
[3] F. Rusek *et al.*, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, Jan. 2013.
[4] H. Van Trees, *Optimum Array Processing, Detection, Estimation, and Modulation Theory (Part IV)*. New York, NY, USA: Wiley, 2002.
[5] L. Jin, S. Li, and B. Hu, "RNN models for dynamic matrix inversion: A control-theoretical perspective," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 189–199, Jan. 2018.
[6] X.-W. Zhang, L. Zuo, M. Li, and J.-X. Guo, "High-throughput FPGA implementation of matrix inversion for control systems," *IEEE Trans. Ind. Electron.*, early access, May 28, 2020, doi: 10.1109/TIE.2020.2994865.
[7] J. L. Jerez *et al.*, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3238–3251, Dec. 2014.
[8] *Xilinx Virtex-7*. Accessed: 2017. [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html
[9] S. D. Munoz and J. Hormigo, "High-throughput FPGA implementation of QR decomposition," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 9, pp. 861–865, Sep. 2015.
[10] S. Chan and X. Yang, "Improved approximate QR-LS algorithms for adaptive filtering," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 51, no. 1, pp. 29–39, Jan. 2004.
[11] R.-H. Chang, C.-H. Lin, K.-H. Lin, C.-L. Huang, and F.-C. Chen, "Iterative QR decomposition architecture using the modified Gram–Schmidt algorithm for MIMO systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 5, pp. 1095–1102, May 2010.
[12] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," in *Proc. Signal Process. Algorithms Archit. Arrangements*, 2013, pp. 70–72.
[13] C. Zhang *et al.*, "On the low-complexity, hardware-friendly tridiagonal matrix inversion for correlated massive MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6272–6285, Jul. 2019.
[14] Y.-W. Xu, Y. Xi, J. Lan, and T.-F. Jiang, "An improved predictive controller on the FPGA by hardware matrix inversion," *IEEE Trans. Ind. Electron.*, vol. 65, no. 9, pp. 7395–7405, Sep. 2018.
[15] Y.-X, Zeng, "A new method for the adjoint matrix of triangular matrices," *J. Tianjin Univ. Finance Econ.*, vol. S1, pp. 102–106, Jan. 1994.
[16] D.S. Watkins, *Fundamentals of Matrix Computations*, 2nd ed. Hoboken, NJ, USA: Wiley, 2002.