Comparative Analysis of Hardware Performance for Matrix Inversions on FPGA Using the Vivado HLS Tool

Nurulhuda binti Ismail[1], Mohamad Hairol bin Jabbar[2], Ariffuddin bin Joret[3], Norshidah binti Katiran[4], Eddy Irwan Shah bin Saadon[5]
[1, 2, 3,4]Department of Electronic Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia.
[4]Department of Electrical Engineering, Center for Diploma Studies, Universiti Tun Hussein Onn Malaysia, Malaysia.

Article Info

Article history:

Received Jan 2, 2024
Revised May 22, 2024
Accepted Aug 24, 2024

Keywords:

Massive MIMO
Linear Detection
Matrix inversion
FPGA
Vivado HLS

ABSTRACT

This paper compares the performance of hardware implementation for linear detection in a massive MIMO system. The study focuses on Gram matrix inversion solved using two approaches: direct and indirect matrix inversion. Direct matrix inversion is represented by Cholesky Decomposition, while indirect matrix inversion is represented by the Neumann series (NS) and the Gauss-Seidel (GS) method. Based on software implementation using Matlab done previously, it is clear that the Cholesky algorithm has better performance in terms of bit error rate (BER) compared to NS and GS algorithms. However, the hardware performance of the algorithms is yet to be evaluated to consider architectural trade-offs to meet application requirements. The algorithm for inversions, embedded in a C-based function, is virtually implemented on the FPGA using the Xilinx Vivado HLS tool. The synthesis report generated form the simulation categorizes the performance from the FPGA implementation into three parts: timing (ns), cycle latency, and resource utilization. With the same targeted time limit, indirect matrix inversion, such as the Neumann series, seems to be the fastest algorithm compared to the direct method due to the matrix-matrix multiplication approach. In terms of latency, NS requires more clock cycles to obtain the output compared to others. Based on the results, the direct inversion method exhibits higher complexity, particularly in timing for clock frequency and resource utilization needed to complete the inversion.

Corresponding Author:

Nurulhuda binti Ismail,
Department of Electronic Engineering,
Faculty of Electrical and Electronic Engineering,
Universiti Tun Hussein Onn Malaysia,
86400, Pt. Raja, Batu Pahat, Johor, Malaysia.
Email: nhuda@uthm.edu.my

1. INTRODUCTION

Matrix inversion is widely used in various fields, particularly in electrical and electronic engineering, including power systems [1], networking [2], control systems [3][4], signal processing [5] and so on. In wireless communication system [6], matrix inversion is applied in the massive MIMO system, specifically in detection and precoding techniques. In the uplink channel, linear detection techniques, such as Zero Forcing (ZF), can perform very well and nearly optimal. As the number of base station antennas (*M*) and the number of users (*K*) within the cell increase, the size of the Gram matrix inversion ($\mathbf{Z}^{-1}$), in the ZF detector becomes significantly larger. Thus, it grows the complexity of the massive MIMO system. Conventional direct matrix inversion methods like Cholesky Decomposition and QR Decomposition become more intricate to do the inversion. Both methods depend mainly on the decomposition process, where the inverted matrix will be decomposed into a product of simple matrices. QR method is introduced due to its stability, and it is applicable for any type of matrices, while Cholesky Decomposition is basically applied for positive definite and non-singular $K \times K$ matrices. Unfortunately, the computational complexity of these methods is still high $O(K^3)$ and difficult for hardware implementation.

The indirect inversion method based on approximate inversion is more popular among the large-scale integrated (VLSI) community. It is an effective tool to compute matrix inversion which is based on matrix multiplications and additions. This technique is generally less complex, and it can be more accurate especially when it is implemented on parallel machines. One way to calculate $\mathbf{Z}^{-1}$ is by computing the solution of the linear system of the equations $\mathbf{Zx} = \mathbf{b}$. There are two types of indirect inversion methods; series expansion method such as Neumann series expansion (NS) [7] and Taylor expansion while iterative methods such as Gauss-Seidel (GS) [8] and Jacobi [9] method. The performance of these inversion approaches in software implementation in terms of bit error rate (BER) has been investigated in [10], [11]. The simulation findings demonstrate that the indirect matrix inversion methods achieve comparable performance to the direct inversion approach, using fewer iterations and having a lower computational complexity.

Implementation of the matrix inversion algorithm onto the hardware should go through the design process. The conventional hardware design process is divided into two primary stages. Firstly, the algorithm's functional description, written in a high-level language, must be manually converted into the Register Transfer Level (RTL). This conversion provides a detailed representation of the algorithm's behaviour during each clock cycle. Logic synthesis is an automated method that transforms the RTL abstraction into a physical hardware design. Transforming the algorithm into RTL is a time-consuming, expensive, and error-prone process. With advanced technology in designing FPGA, High-Level Synthesis (HLS) tools are introduced as a low-time consumption method that can overcome the constraints of the conventional design process. HLS tools are used to systematically transform a specification at a high abstraction level, e.g., algorithmic description into an application-specific architecture, usually implemented as a digital system that is supported by RTL [12]. Xilinx Vivado HLS tool is a commercial compiler developed by Xilinx Vivado HLS tools provide services which support all steps taken in FPGA design, beginning with design entry, simulation, synthesizing, place and route, generating bitstream, debugging, verification, and software development onto targeted FPGA family.

The main contribution of this work is to examine the efficiency of hardware implementation in inverting the Gram matrix. The implementation prioritizes the usage of Cholesky Decomposition for the direct inversion approach, while employing the Neumann series and Gauss-Seidel for the indirect inversion method. Xilinx Vivado HLS tool is used to generate an RTL design of the inversion algorithm that is written in a programming language in C++. The targeted FPGA device for this study is the Kintex-7, Xilinx 7 series FPGA family. This device provides low power consumption, which is one of the primary elements that is of concern to most designers. Having a low-power system, can reduce the cost, improve the reliability of the device, and consequently improve the performance of the system[13]. Once the source code is analysed, Vivado HLS provides a synthesis report that classifies the hardware performance into three parts: timing (ns), cycle latency, and estimation of resource utilization. The performance is then compared between the three algorithms of the matrix inversion as mentioned above.

The rest of this paper is organized as follows: Section 2 discusses the system model of linear detection in the uplink in a massive MIMO. Section 3 presents the matrix inversion solved by direct and indirect methods. It also explains the implementation of the matrix inversion on the FPGA using the Xilinx Vivado HLS tool. The performance of the hardware implementation for both approaches has been compared in Section 4. Section 5 concludes the entire paper.

2. System Model

This section discusses the system model of the uplink channel of the massive MIMO system.

2.1. System model of the uplink channel

In this study, a massive MIMO system with M number of antennas at the base station (BS) and K single-antenna users is considered, where $M \gg K > 1$ as illustrated in Figure 1. For the uplink channel, the K users are simultaneously transmitting bit streams to the BS. Bit-streams from individual users will be encoded separately by the channel encoder. Subsequently, these encoded streams will be mapped into symbols from an energy-normalized modulation constellation, denoted as Ω. Vector $\boldsymbol{s} = [s_1, \ldots, s_K]^T$ represents the transmit symbols for all users. K. The transmit vector is then transmitted over the wireless channel model written as

$$\boldsymbol{y} = \mathbf{Hs} + \mathbf{n}, \tag{1}$$

where $y = [y_1, \ldots, s_K]^T$ is the received vector at the BS, $\mathbf{H} \in C^{M\times K}$ denotes the matrix of Rayleigh Fading channel coefficients between the BS antenna array and the K users. $\mathbf{n} \in C^M$ denotes an additive white Gaussian noise (AWGN) vector with independent and identical distribution (i.i.d), zero mean, unit variance complex Gaussian random variables. The transmitted signals are assumed to be i.i.d Gaussian distribution and it satisfies $E\{|s_i|^2\} = E_s$, for $1 \leq i \leq K$.

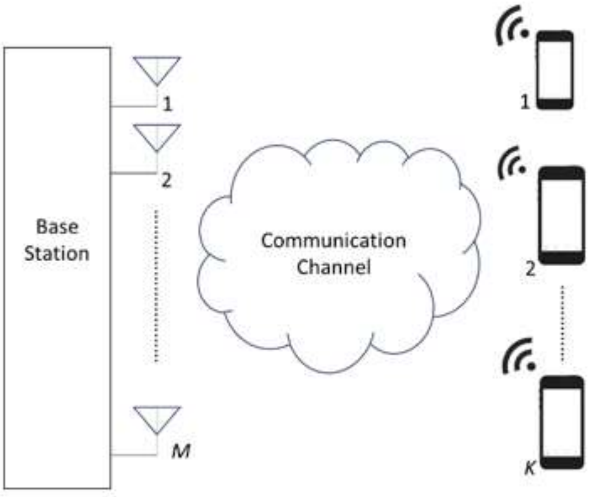

Figure 1. Massive MIMO system model

2.2. Linear Detection for massive MIMO

In massive MIMO systems, the transmitted symbol vector $\mathbf{s}$ is determined by the symbol detector from the received signal vector $\mathbf{y}$. Nonlinear detectors are not very hardware-friendly. For instance, expectation propagation detection (EPD) [14]faces difficulties with parallelism. Other nonlinear detection methods, such as successive interference cancellation (SIC) and lattice reduction aid (LRA) [15], suffer from incredibly high complexity. Therefore, linear detection methods such as Linear Minimum Mean-Square Error (MMSE) Equalization and Zero Forcing (ZF) are commonly applied in massive MIMO as these methods are easier to implement due to less complexity and simpler structure.

For the MMSE detection, the effect of noise is considered. Thus, the transmitted vector can be computed as

$$\tilde{\mathbf{s}}_{MMSE} = (\mathbf{H}^H\mathbf{H} + \boldsymbol{\sigma}^2\mathbf{I}_K)^{-1}\mathbf{H}^H\mathbf{y} \; = \; (\mathbf{Z})^{-1}\mathbf{H}^H\mathbf{y}, \tag{2}$$

where $\mathbf{I}_K$ denoted as the $K \times K$ identity matrix. The Gram matrix is modified with a regularization by noise variance and it is expressed as $\boldsymbol{Z}_{MMSE} = \mathbf{H}^H\mathbf{H} + \boldsymbol{\sigma}^2\mathbf{I}_K$.

Assume that the effect of noise is ignored and the transmitted vector can be computed by ZF from the observed received signal vector $\mathbf{y}$ as

$$\tilde{\mathbf{s}}_{ZF} = \mathbf{H}^+\mathbf{y} = \; (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{y} \; = \; (\mathbf{Z})^{-1}\mathbf{H}^H\mathbf{y}, \tag{3}$$

where $\mathbf{H}^H$ is the transpose and conjugate operation on the channel matrix $\mathbf{H}$. The ZF detector is required to perform Gram matrix inversion where $\boldsymbol{Z}_{ZF}{}^{-1} = (\mathbf{H}^H\mathbf{H})^{-1}$. The multiuser interference is completely eliminated by multiplying the received signal vector, $\mathbf{y}$, with the pseudo-inverse channel matrix as written in equation (3). Figure 2 illustrates the block diagram of the Linear ZF detection.

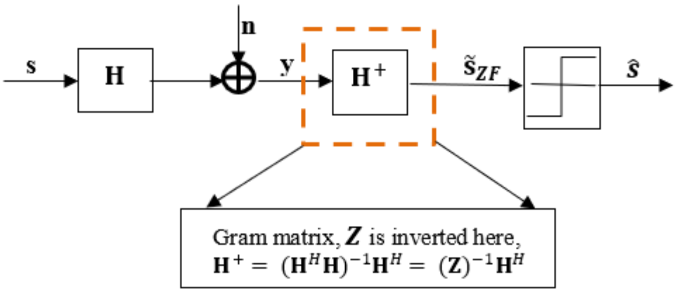
s
H
n
y
$\mathbf{H}^+$
$\tilde{\mathbf{s}}_{ZF}$
$\hat{\mathbf{s}}$
Gram matrix, $\boldsymbol{Z}$ is inverted here,
$\mathbf{H}^+ = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H = (\mathbf{Z})^{-1}\mathbf{H}^H$

Figure 2. Block diagram of ZF Detector

3. RESEARCH METHOD

This section is divided into two parts. The first part is about the algorithm of the inversion method, which can be classified into direct inversion methods and indirect inversion methods. The second part is about the implementation of matrix inversion on the FPGA using the Vivado HLS tool.

3.1. Matrix inversion Method

3.1.1. Direct Matrix Inversion Method

Several direct inversion methods can be applied to solve Gram matrix inversion, $\boldsymbol{Z}^{-1}$ which have been discussed in [16] and [17]. Cholesky Decomposition [18] is one of the direct inversion methods other than QR Decomposition and Gauss Elimination. It is a useful technique for efficient arithmetic operations and numerical solutions of a linear equation. Besides, Cholesky Decomposition is a low complex direct inversion method which can be applied in Very-large-scale-integration (VLSI). This approach is unique for Hermitian Positive Definite Matrix (HPDM) such as matrix **Z** given in equation (3). This method applied a $K \times K$ lower triangular matrix, **L** to decompose a Gram matrix **Z** as $\mathbf{Z} = \mathbf{L}\mathbf{L}^{H}$. Below is the pseudocode for the Cholesky Decomposition.

```
Algorithm  Cholesky Decomposition
Input:
• Gram matrix, $\mathbf{Z} = \mathbf{H}^{H}\mathbf{H}$
• Decompose $\mathbf{Z}$ into lower ($\mathbf{L}$) and upper ($\mathbf{L}^{H}$) triangular matrix
for $j$=1:$K$ do
  • Calculate diagonal elements of $\mathbf{L}$,
    $l_{j,j} = \sqrt{z_{j,j} - \sum_{k=1}^{j-1} l_{j,k} l_{j,k}^{H}}$
    for $i$=1:$j$-1 do
      • Calculate the off-diagonal elements of $\mathbf{L}$,
        $l_{i,j} = \frac{1}{l_{j,j}}\left(z_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k}^{H}\right)$ for $i > j$
    end for
end for
  • Obtain the lower triangular matrix, $\mathbf{L}$
  • Calculate $\mathbf{L}^{-1}$ based on $\mathbf{L}\mathbf{L}^{-1} = \mathbf{I}$
  • Obtain $(\mathbf{L}^{-1})^{H}$
Output: $\mathbf{Z}^{-1} = (\mathbf{L}^{-1})^{H}\mathbf{L}^{-1}$
```

3.1.2. Indirect Matrix Inversion Method

The inversion of **Z** can also be addressed using indirect methods. In the study of massive MIMO detection, two types of approximate matrix inversion methods are commonly employed today: i) Series Expansion Methods such as Neumann series expansion and Taylor expansion, and ii) iterative methods like Gauss-Seidel and Jacobi methods.

i) Neumann series expansion method

Gram matrix inversion can be solved using the Neumann series by replacing matrix inversion with matrix multiplication as written below

$$\mathbf{Z}^{-1} = \sum_{n=0}^{N-1} (\mathbf{I_K} - \theta^{-1}\mathbf{Z})^{n}\mathbf{X}^{-1} \tag{4}$$

where $\mathbf{X}^{-1}$ is an arbitrary matrix that holds the convergence condition of $lim_{n\to\infty}(\mathbf{I_K} - \theta^{-1}\mathbf{Z})^{n} = 0_{K}$.

ii) Gauss-Seidel method

In Gauss-Seidel method, the Gram matrix is decomposed into three main components; diagonal (**D**), strictly upper triangular matrix (**U**), and strictly lower triangular matrix (**L**) represented as $\mathbf{Z} = \mathbf{U} + \mathbf{D} + \mathbf{L}$. By applying linear equation, the estimated transmitted signal vector can be computed as

$$\tilde{\boldsymbol{x}}^{(\boldsymbol{i}+\mathbf{1})} = (\mathbf{D} + \mathbf{L})^{-\mathbf{1}}(\boldsymbol{x} - \boldsymbol{U}\tilde{x}^{\boldsymbol{i}}), \qquad i = 0,1,2, \dots.. \tag{5}$$

iii) Jacobi method

By using Jacobi method, the estimated transmitted signal vector is expressed as

$$\tilde{\boldsymbol{x}}^{(\boldsymbol{i}+\mathbf{1})} = \mathbf{D}^{-\mathbf{1}}(\boldsymbol{x} - \mathbf{R}\tilde{x}^{\boldsymbol{i}}), \qquad i = 0,1,2, \dots. \tag{6}$$

where **D** denotes a diagonal matrix while **R** is an off-diagonal matrix given as $\mathbf{R} = \mathbf{Z} - \mathbf{D}$. $\tilde{x}^{\boldsymbol{i}}$ is the solution vector at the *i*-th iteration.

3.2. Hardware Implementation using Vivado HLS Tool

In this part, the inversion of the Gram matrix **Z** using direct and indirect methods is implemented on the virtual FPGA board using a Xilinx tool known as Vivado HLS. With advanced technology in designing FPGA, High-Level Synthesis (HLS) tools are introduced as a low-time consumption method. HLS tools are used to systematically transform a specification at a high abstraction level, e.g., algorithmic description into an application-specific architecture, usually implemented as a digital system that is supported by RTL [12]. FPGA implementation using high-level descriptions is more compact, and it has fewer errors. HLS is used to compile the C function of the matrix inversion algorithm into the RTL description. Figure 3 demonstrates the flowchart on how the Xilinx Vivado HLS tool converts from the C++ function of the inversion method into the hardware description.

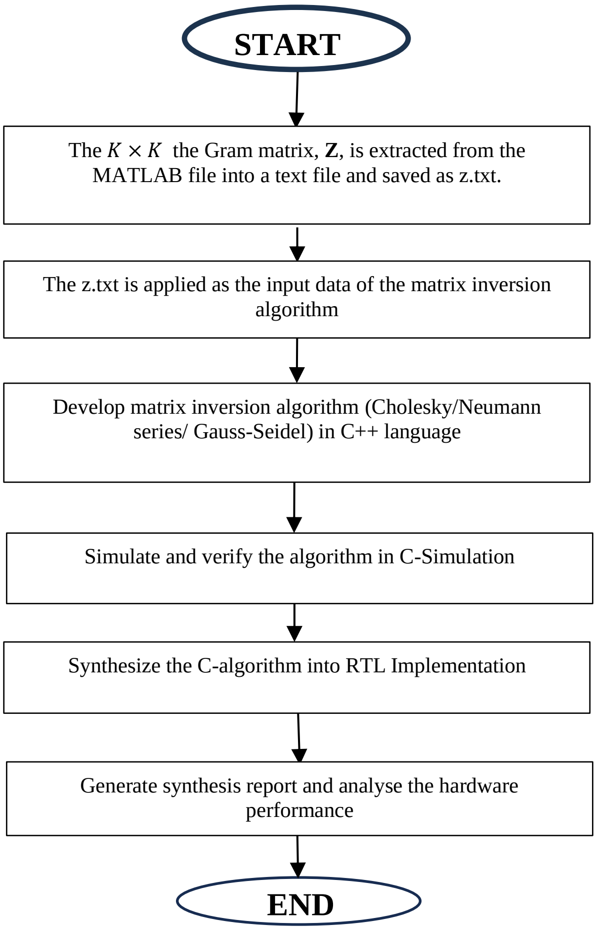
START
The $K \times K$ the Gram matrix, **Z**, is extracted from the MATLAB file into a text file and saved as z.txt.
The z.txt is applied as the input data of the matrix inversion algorithm
Develop matrix inversion algorithm (Cholesky/Neumann series/ Gauss-Seidel) in C++ language
Simulate and verify the algorithm in C-Simulation
Synthesize the C-algorithm into RTL Implementation
Generate synthesis report and analyse the hardware performance
END

Figure 3. Process to convert from C-function into hardware description using Vivado HLS tool

In Xilinx Vivado HLS, the C-based algorithm is separated into the Testbench File and Source File. The transformation in HLS begins with the testbench files, which consist of a testbench function, the $K \times K$ Gram matrix**,** $\mathbf{Z}$, and gold data. The Gram matrix is generated from the random Rayleigh Fading Channel in a massive MIMO system using MATLAB software. It is saved as a text file and transformed into a testbench file so that it can be the input of the C-based algorithm.

In addition, the Source file consists of a header function and a top-level function. The header function is used to define the variables such as the number of rows and columns, the value of the normalization factor which is needed in the Neumann series algorithm, and the number of maximum iterations for the Neumann Series and Gauss-Seidel algorithms. The top-level function that implies all the operations used to obtain $\mathbf{Z}^{-1}$ is then compiled and executed in the C-Simulation. The simulation result is verified with the gold data in the test bench file to ensure that all the functions in the top-level function can be synthesized without error.

Vivado HLS then synthesize the C-algorithm into an RTL implementation that meets the specified timing, performance, and also resource requirements. Vivado generates a comprehensive synthesis report consisting of information related to performance metrics such as area, latency, initiation interval (II), loop iteration latency, loop initiation interval, and loop latency. The report analyses whether the design meets the specified requirements or not. Then, the performance of the Cholesky Decomposition algorithm is compared with indirect matrix inversion algorithms such as Neumann series and Gauss-Seidel.

4. RESULTS AND DISCUSSION

This section compares the results of the direct and indirect matrix inversion algorithms that are implemented on the Xilinx Kintex-7 xc7k70t-fbv676-1.

4.1. Synthesis report

The synthesis report shows the general information about the project, such as the version of the software that is used in the project, the name of the project, the name of the solution, and the technology details, including the product family of Xilinx FPGA and also the targeted device of the selected product. Besides that, the report also produces performance estimates consisting of Timing, Latency, and Utilization Estimates.

Timing shows the target clock frequency and clock uncertainty, and it also estimates the fastest achievable clock frequency. Latency describes the number of cycles that the design needs to produce the output while the interval shows the number of clock cycles before the new inputs can be applied. The latency required to execute all the iterations of the loop for the algorithm is indicated by the max and min latencies. The synthesis report also provides the estimated resources used to implement the design, including look-up-table (LUTS), Flip-Flops, BRAM-18K, and DSP48s. Below is the summary of the synthesis report for three matrix inversion algorithms: Cholesky Decomposition as a direct matrix inversion, Neumann series and Gauss-Seidel as an indirect matrix inversion algorithm.

4.1.1. Timing

Table 1 illustrates the estimated time (nsec) required by the three inversion methods for 8×8 , 16×16 and 32×32 size of the Gram matrix to do inversion. By default, the estimated timing in Vivado HLS is set to be 10 nsec. Among the three methods, Cholesky Decomposition requires more clock frequency to process the inversion compared to the Gauss-Seidel and Neumann series expansion method. This is due to a higher processing requirement by the direct inversion method. This algorithm needs more data to be processed within the same time constraints.

On the other hand, the Neumann series needs the lowest clock frequency because this algorithm is less complex as it requires only multiplications and additions to do the inversion.

Table 1. Estimated Time (ns)

Antenna Configuration	$\mathbf{8 \times 8}$			$\mathbf{16 \times 16}$			**32 x 32**		
Inversion Algorithm	**Cholesky**	**NS**	**GS**	**Cholesky**	**NS**	**GS**	**Cholesky**	**NS**	**GS**
Estimated Time (nsec)	9.93	8.39	9.01	9.83	8.37	9.00	9.84	8.25	8.46

4.1.2. Latency

Table 2 illustrates the number of cycles each algorithm required to produce the result of the inversion. Latency is the number of clock cycles that are required to compute all output values from a given set of inputs. Based on the reading, the Neumann series requires more cycles to obtain the result of inversion compared to the Cholesky Decomposition and Gauss-Seidel. For 8 × 8 antenna configuration, NS requires 5 times higher than GS, and 6 times higher than Cholesky, while for 16 × 16 antenna configuration, NS requires 8 times higher than GS and 15 times higher than Cholesky. When the matrix size increases to 32 × 32, NS remains the highest while Cholesky needs the minimum number of clock cycles to do inversion.

Table 2. Latency (cycle)

Antenna Configuration	**8 × 8**			**16 × 16**			**32 x 32**		
Inversion Algorithm	**Cholesky**	**NS**	**GS**	**Cholesky**	**NS**	**GS**	**Cholesky**	**NS**	**GS**
Latency(cycle)	5148	34961	6723	25765	396823	45187	50917	5636012	135318

4.1.3. Utilization Estimates

Table 3 illustrates the resource utilization required by these three methods. There is notable saving in logic slices, DSP48 and also BRAM. Based on the resources provided for the targeted device as listed in the first column, the resources utilized by each inversion method are sufficient. For the size of 8 × 8, within 5 number of iterations, the Neumann series requires very little resource utilization with 0% for BRAM_18K, 7% for DSP48E and LUT and only 1% for FF while Cholesky requires very high resources utilization; 4% for BRAM_18K, 31% for DSP48E, 12% for FF and 24% for LUT. The reading of Gauss-Seidel is in between these two algorithms. The pattern of performance for each method remains the same even though the size of the Gram matrix increased. According to the author in [19], the hardware implementation for the Neumann series and Gauss-Seidel methods is evaluated for 8 users using Xilinx Virtex-7 XC7VX690T FPGA. The result shows that the Neumann series requires 148797 LUTs, 161934 FFs, and 1016 DSP48s while Gauss-Seidel requires 18976 LUTs, 15864 FFs, and 232 DSP48s. The architecture is able to reach different application requirements by adjusting the number of iterations as required. This implementation shows its advantages in hardware efficiency and flexibility.

Table 3. Resources utilization

Antenna Configuration	**8 × 8**			**16 × 16**			**32 x 32**		
Inversion algorithm	**Cholesky**	**NS**	**GS**	**Cholesky**	**NS**	**GS**	**Cholesky**	**NS**	**GS**
BRAM_18K **(270)**	12 (4%)	0 (0%)	0 (0%)	16 (6%)	0 (0%)	0 (0%)	75 (28%)	0 (0%)	0 (0%)
DSP48E **(240)**	75 (31%)	16 (7%)	24 (10%)	75 (31%)	16 (7%)	24 (10%)	75 (31%)	16 (7%)	24 (10%)
FF **(82000)**	10052 (12%)	2030 (1%)	3214 (3%)	10013 (12%)	2225 (3%)	3962 (5%)	9844 (12%)	2216 (3%)	3466 (4%)
LUT **(41000)**	10092 (25%)	2681 (7%)	5012 (12%)	10190 (25%)	2745 (7%)	5117 (13%)	10230 (25%)	2752 (7%)	5077 (12%)

5. CONCLUSION

In conclusion, the performance of ZF-based massive MIMO detection in a hardware implementation has been compared. The comparison is based on the Gram matrix inversion using Cholesky Decomposition as a direct inversion method and Neumann series and Gauss-Seidel as an indirect inversion method for 8 × 8, 16 × 16 and 32 × 32 antenna configurations. Indirect methods improve both estimated time and hardware usage. In terms of resource utilization, indirect methods are significantly reduced, especially BRAM and also DSP resources. However, the Neumann series requires a clock frequency five times higher than the Cholesky and Gauss-Seidel methods. It is clear from the results that although Cholesky has better performance in software implementation, GS and NS algorithms show better performance in hardware implementation. The direct inversion method exhibits higher complexity, particularly in timing for clock frequency and resource utilization that are needed to complete the inversion. An experimental setup to investigate and compare the hardware performance of the inversion matrix with the RTL or other methods is suggested as a future work.

ACKNOWLEDGMENTS

The authors thank the Universiti Tun Hussein Onn Malaysia for financing the research project under the TIER 1 Grant.

REFERENCES

[1] M. O. Faruque *et al.*, “Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis,” *IEEE Power Energy Technol. Syst. J.*, vol. 2, no. 2, pp. 63–73, 2015, doi: 10.1109/JPETS.2015.2427370.

[2] T. Ould-Bachir, H. F. Blanchette, and K. Al-Haddad, “A network tearing technique for FPGA-based real-time simulation of power converters,” *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3409–3418, 2015, doi: 10.1109/TIE.2014.2365752.

[3] W. P. Hunek, “New SVD-based matrix H-inverse vs. minimum-energy perfect control design for state-space LTI MIMO systems,” *2016 20th Int. Conf. Syst. Theory, Control Comput. ICSTCC 2016 - Jt. Conf. SINTES 20, SACCS 16, SIMSIS 20 - Proc.*, no. 1, pp. 14–19, 2016, doi: 10.1109/ICSTCC.2016.7790633.

[4] 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.*, vol. 68, no. 7, pp. 6205–6216, 2021, doi: 10.1109/TIE.2020.2994865.

[5] A. Chakraborty and A. Banerjee, “A Multiplierless VLSI architecture of QR decomposition based 2D wiener filter for 1D/2D signal processing with high accuracy,” *2018 4th Int. Conf. Comput. Commun. Autom. ICCCA 2018*, pp. 1–6, 2018, doi: 10.1109/CCAA.2018.8777458.

[6] T. L. Marzetta, “Massive MIMO: An Introduction,” *Bell Labs Tech. J.*, vol. 20, pp. 11–22, 2015, doi: 10.15325/BLTJ.2015.2407793.

[7] H. Prabhu, J. Rodrigues, O. Edfors, and F. Rusek, “Approximative matrix inverse computations for very-large MIMO and applications to linear pre-coding systems,” in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 2710–2715. doi: 10.1109/WCNC.2013.6554990.

[8] X. Gao, L. Dai, J. Zhang, S. Han, C. L. I, and I. Chih-Lin, “Capacity-approaching linear precoding with low-complexity for large-scale MIMO systems,” *IEEE Int. Conf. Commun.*, vol. 2015-Septe, pp. 1577–1582, Jun. 2015, doi: 10.1109/ICC.2015.7248549.

[9] J. Minango and C. de Almeida, “A Low-Complexity Linear Precoding Algorithm Based on Jacobi Method for Massive MIMO Systems,” in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, Jun. 2018, pp. 1–5. doi: 10.1109/VTCSpring.2018.8417575.

[10] F. Wang *et al.*, “Efficient Matrix Inversion Architecture for Linear Detection in Massive MIMO Systems,” *2015 IEEE Int. Conf. Digit. Signal Process.*, pp. 248–252, 2015, doi: 10.1109/ICDSP.2015.7251869.

[11] C. Zhang, Z. Wu, S. Member, C. Studer, and S. Member, “Efficient Soft-Output Gauss – Seidel Data Detector for Massive MIMO Systems,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. PP, pp. 1–12, 2018, doi: 10.1109/TCSI.2018.2875741.

[12] D. Koch, F. Hannig, and D. Ziener, *FPGAs for software programmers*. Springer, 2016. doi: 10.1007/978-3-319-26408-0.

[13] Xilinx, “Xilinx 7 series fpgas: breakthrough power and performance, dramatically reduced development time,” 2012.

[14] X. Tan, Y. L. Ueng, Z. Zhang, X. You, and C. Zhang, “A Low-complexity massive mimo detection based on approximate expectation propagation,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7260–7272, 2019, doi: 10.1109/TVT.2019.2924952.

[15] Y. H. Gan, C. Ling, and W. H. Mow, “Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection,” *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2701–2710, 2009, doi: 10.1109/TSP.2009.2016267.

[16] A. Jennings, *Matrix Computation for Engineers and Scientists*. John Wiley and Sons, 1977.

[17] G. Stewart, *Matrix Algorithm: Basic Decompositions.* Philadelphia, PA: SIAM, 1998.

[18] A. Krishnamoorthy and D. Menon, “Matrix Inversion Using Cholesky Decomposition,” *CoRR*, vol. abs/1111.4, no. 3, pp. 70–72, 2011.

[19] Z. Wu, C. Zhang, Y. Xue, S. Xu, and X. You, “Efficient architecture for soft-output massive MIMO detection with Gauss-Seidel method,” *Proc. - IEEE Int. Symp. Circuits Syst.*, vol. 2016-July, pp. 1886–1889, 2016, doi: 10.1109/ISCAS.2016.7538940.

BIOGRAPHY OF AUTHORS

Nurulhuda binti Ismail graduated from Universiti Teknologi Mara (UiTM) Shah Alam with a Bachelor's degree in Electrical & Electronic Engineering (Telecommunications) in 2004. Subsequently, she obtained her Master's in Electrical Engineering in 2007 from Universiti Tun Hussein Onn Malaysia. Presently, she holds a position as a lecturer at Universiti Tun Hussein Onn Malaysia (UTHM) and is affiliated with the Advanced Telecommunication Research Centre (ATRC) at the same institution. Nurulhuda's research interests encompass digital signal processing, precoding, massive MIMO, 5G, and beyond. Her email: nhuda@uthm.edu.my

Mohamad Hairol Bin Jabbar completed his PhD from the University of Grenoble, France. He has been a senior lecturer since 2013 at the Faculty of Electronic and Electrical Engineering, Universiti Tun Hussein Onn Malaysia (UTHM). His research interests include multiprocessor system-on-chip (MPSOC) architecture, embedded systems design and parallel processing. His email: hairol@uthm.edu.my

Norshidah binti Katiran is a distinguished academic with a Bachelor of Electrical Engineering (Telecommunications) degree from Universiti Teknologi Malaysia (UTM) in 2001 and a Ph.D. in Electrical Engineering from the same university in 2015. Currently serving as a senior lecturer at Universiti Tun Hussein Onn Malaysia (UTHM), she is also an esteemed affiliate researcher at the Advanced Telecommunication Research Centre (ATRC), Universiti Tun Hussein Onn Malaysia. Norshidah's research focuses on resource allocation for wireless networks and non-orthogonal multiple access (NOMA) for 5G and beyond. Her email: norshida@uthm.edu.my

Ariffuddin bin Joret is a senior lecturer at the Faculty of Electrical and Electronic Engineering (FKEE), Universiti Tun Hussein Onn Malaysia (UTHM). He obtained his first degree in Electrical and Electronics Engineering from Universiti Teknologi Mara (UiTM). Subsequently, he completed his Master's Degree at Universiti Sains Malaysia (USM) in 2006, specializing in Master of Science (Electrical and Electronics Engineering) with a focus on Neural Networks. In 2018, he graduated with a Ph.D. in signal processing from UTHM. His current research interests encompass Digital Signal Processing, Digital Modulation and Communication Systems, Ground Penetrating Radar, and the development and application of Internet of Things (IoT) systems. Presently, he is a researcher at the Internet of Things Focus Group (IoT FG), FKEE. He can be contacted at email: ariff@uthm.edu.my

Eddy Irwan Shah bin Saadon is a dedicated lecturer at the Centre for Diploma Studies, Universiti Tun Hussein Onn Malaysia (UTHM). His journey in academia is marked by a strong foundation in Electrical Communication Engineering, having graduated with honors from Universiti Teknologi Malaysia in 2001. Motivated by a passion for advancing his expertise, he subsequently pursued a Master's degree in Electrical Engineering at Universiti Tun Hussein Onn Malaysia, completing his studies in 2013. His expertise is in digital communication, specifically in ultra-wideband. His email: eddy@uthm.edu.my