Inversión Aproximada de Matrices en Sistemas Massive MIMO Correlados en Tiempo o Frecuencia

Óscar Cobos Morales, Rafael Soldado Guerrero, Eduardo Martos Naya, Francisco Javier López Martínez, José Tomás Entrambasaguas Muñoz
ocm@ic.uma.es, rsg@ic.uma.es, eduardo@ic.uma.es, fjlopezm@ic.uma.es, jtem@ic.uma.es
Dpto. de Ingeniería de Comunicaciones. Universidad de Málaga. 29071

***Abstract*- Massive multiple-input multiple-output (MIMO) is expected to be one of the keys in 5G. In this technology, the base station is equipped with a big number of antennas serving multiple users simultaneously to improve spectral efficiency, coverage, and range. Zero-Forcing and Minimum Mean Square Error have been considered potential practical precoding and detection methods for large scale MIMO systems but require much larger dimensions of matrix inversion. This paper presents an architecture for approximate matrix inversion based on Neumann Series, thereby reducing the cost of hardware. In addition, we propose a solution for systems with time or frequency correlation among different channels where we are able to reach a much higher throughput.**

Index Terms—FPGA, Low complexity, Massive MIMO, matrix inversion, Neumann Series, Time- frequency correlation.

I. INTRODUCCIÓN

La tecnología *Massive MIMO* se ha convertido en un tema de interés en la quinta generación de sistemas de comunicaciones móviles, ya que aporta una mejora en cuanto a eficiencia espectral, fiabilidad del enlace y cobertura. Esta técnica se basa en utilizar cientos de antenas en la estación base *(BS)*, para dar servicio a decenas de usuarios en el mismo recurso tiempo-frecuencia [3]. La desventaja de esta tecnología es que aumenta la complejidad de la *BS*.

En los sistemas *Massive MIMO*, se usan métodos de detección como *Zero-Forcing* (*ZF*) o *Minimum Mean Square Error* (*MMSE*) para mantener un equilibrio entre rendimiento y complejidad. Sin embargo, estas técnicas requieren invertir matrices, lo que supone elevar el coste computacional y usar más recursos hardware en *Application-Specific Integrated Circuits (ASIC).* En este trabajo se propone una arquitectura para invertir matrices que reduzca estos problemas.

Los métodos de inversión de matrices se pueden dividir en dos familias: métodos directos y métodos iterativos. Los métodos directos encuentran la solución en un paso y los métodos iterativos refinan la solución en cada iteración. Algunos ejemplos de métodos directos serían las descomposiciones QR [6], [7], LU [8], [9] y Cholesky [10], [11], mientras que algunos algoritmos iterativos serían los de Gradiente Conjugado [13], [14], de Gauss-Seidel [4], [12] y las series de Neumann [1], [2], [3], [6]. Además, estos métodos se pueden clasificar en métodos exactos y métodos aproximados. Los métodos exactos, aunque tienen un alto coste computacional, ofrecen resultados con mayor exactitud. Por el contrario, los métodos aproximados son más rápidos y menos complejos, pero los resultados son menos exactos.

En este trabajo, se ha optado por las series de Neumann para la inversión de matrices. Se trata de un método iterativo y aproximado que ofrece un compromiso entre rendimiento y complejidad. Se propone una nueva técnica para escenarios donde existe una correlación en tiempo o frecuencia entre las respuestas del canal. Las series de Neumann emplean esta relación para obtener la matriz inversa a partir de inversiones anteriores, ahorrando tiempo de cálculo y recursos hardware.

En [1], se propone una arquitectura similar a la planteada, usando series de Neumann para invertir matrices en sistemas *Massive MIMO*. La diferencia está en que en [1] se suponen matrices del canal de diagonal dominante y en este trabajo matrices correladas en tiempo o frecuencia. En la práctica es común encontrar este tipo de escenarios, sin embargo el supuesto de [1] no siempre se cumple, por ejemplo cuando el número de antenas en la *BS* no es muy elevado.

Este documento se estructura como sigue. La sección II describe el modelo de sistema para un escenario *Massive MIMO*. En la sección III se explica el método basado en las series de Neumann y se propone una arquitectura para la inversión matricial. La sección IV plantea la implementación hardware del inversor. La sección V incluye un estudio comparativo de la complejidad del algoritmo y analiza los resultados y prestaciones obtenidas tras implementar el diseño en una FPGA. Por último, la sección VI presenta las conclusiones discutiendo el sistema desarrollado.

II. MODELO DE SISTEMA PARA MASSIVE MIMO

En la sección I, se ha descrito *Massive MIMO* como una técnica en la que hay múltiples antenas que sirven a varios usuarios. Sea N el número de antenas en una *BS* y $M(< N)$ el número de usuarios servidos por cada antena. El modelo de sistema para *Massive MIMO* viene dado por (1):

$$\bar{y} = \boldsymbol{H}\bar{s} + \bar{n} \tag{1}$$

donde $\bar{s} = [s_1, s_2, \dots, s_M]^T$ es el vector transmitido, $\bar{y} = [y_1, y_2, \dots, y_N]^T$ es el vector recibido, $\bar{n}$ es el vector de ruido y $\boldsymbol{H}$ es la matriz del canal (compleja de tamaño $M\ x\ N$).

Para recuperar los datos transmitidos $\bar{s}$, el vector recibido $\bar{y}$ se ha de multiplicar por una de las matrices de detección propuestas en los algoritmos *ZF* o *MMSE*, [15]:

$$\boldsymbol{W}_{ZF} = \boldsymbol{H}^H(\boldsymbol{H}\boldsymbol{H}^H)^{-1} \tag{2}$$

$$\boldsymbol{W}_{MMSE} = \boldsymbol{H}^H(\boldsymbol{H}\boldsymbol{H}^H + \sigma\boldsymbol{I})^{-1} \tag{3}$$

donde $\boldsymbol{I}$ es la matriz identidad, σ es la potencia de ruido y el superíndice $(\cdot)^H$ denota el conjugado traspuesto.

En (2) y (3), la matriz compleja $\boldsymbol{H}\boldsymbol{H}^H$ es de tamaño $M\ x\ M$. Aunque el número de usuarios no sea demasiado elevado, la inversión de matrices es una operación costosa.

En este trabajo, se propone el algoritmo de inversión iterativo y aproximado basado en series de Neumann para

sistemas *Massive MIMO* donde las matrices del canal están correladas en tiempo o frecuencia. Este método pretende reducir la complejidad a la hora de realizar la inversión de matrices en los algoritmos de detección *ZF* y *MMSE*.

En este punto, las matrices $\boldsymbol{HH}^H$ y $\boldsymbol{HH}^H + \sigma \boldsymbol{I}$ a invertir en los algoritmos *ZF* y *MMSE* pasarán a denominarse $\boldsymbol{A}$.

III. Arquitectura propuesta para inversión de matrices basada en series de Neumann

A. Aproximación por series de Neumann

En este apartado, se describe cómo se invierte una secuencia de matrices del canal con correlación en tiempo o frecuencia usando la aproximación por series de Neumann.

Si la matriz $\boldsymbol{A}$ es parecida a una matriz invertible $\boldsymbol{B}$, tal que:

$$\lim_{n\to\infty} (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})^n = 0 \tag{4}$$

la inversa de $\boldsymbol{A}$ se puede expresar con series de Neumann como sigue, [1]:

$$\boldsymbol{A}^{-1} = \sum_{n=0}^{\infty} (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})^n \boldsymbol{B}^{-1} \tag{5}$$

Si se trunca la expresión (5) a los k primeros términos y se saca factor común $\boldsymbol{B}^{-1}$ la inversa aproximada sería:

$$\boldsymbol{A}^{-1} \approx \boldsymbol{A}_k^{-1} = \sum_{n=0}^{k-1} (\boldsymbol{B}^{-1}(\boldsymbol{B} - \boldsymbol{A}))^n \boldsymbol{B}^{-1}, k \geq 1. \tag{6}$$

Sean $\boldsymbol{A}_i$ las matrices del canal a invertir correladas a lo largo del tiempo o la frecuencia, donde el subíndice i se refiere al índice temporal o frecuencial. La matriz $\boldsymbol{A}_{i+1}$ se puede descomponer como aparece en la expresión (7):

$$\boldsymbol{A}_{i+1} = \boldsymbol{A}_i + \boldsymbol{E}_{i+1} \tag{7}$$

donde la matriz $\boldsymbol{E}_{i+1}$ es la diferencia entre la nueva matriz a invertir y la matriz previamente invertida.

Sustituyendo $\boldsymbol{A} = \boldsymbol{A}_{i+1}$ y $\boldsymbol{B} = \boldsymbol{A}_i$ en la expresión (6), se puede obtener la nueva matriz inversa a partir de la anterior:

$$\boldsymbol{A}_{i+1,k}^{-1} = \sum_{n=0}^{k-1} (-\boldsymbol{A}_i^{-1}\boldsymbol{E}_{i+1})^n \boldsymbol{A}_i^{-1}, k \geq 1. \tag{8}$$

En la expresión (8) se observa que la inversión se realiza sólo con productos y sumas de matrices. Por este motivo, el algoritmo propuesto tiene una complejidad más baja y usa menos recursos hardware que otros métodos. Controlando el número de iteraciones, se puede obtener mayor exactitud a cambio de aumentar la complejidad.

B. Arquitectura propuesta del inversor matricial

En este apartado se presenta la arquitectura basada en series de Neumann para cualquier valor de *k*.

Con el objetivo de simplificar, de aquí en adelante se usa la siguiente notación: $\boldsymbol{A} = \boldsymbol{A}_{i+1}$, $\boldsymbol{B} = \boldsymbol{A}_i$ e $\boldsymbol{Y} = \boldsymbol{B}^{-1} = \boldsymbol{A}_i^{-1}$ para las matrices de entrada. Además, se consideran las matrices intermedias $\boldsymbol{E} = \boldsymbol{E}_{i+1}$, $\boldsymbol{X} = -\boldsymbol{A}_i^{-1}\boldsymbol{E}_{i+1}$, $\boldsymbol{Z}_k = \boldsymbol{A}_{i+1,k}^{-1}$ y $\boldsymbol{R} = \boldsymbol{Z}_{k-1} = \boldsymbol{A}_{i+1,k-1}^{-1}$. De esta forma la expresión (8) se puede reescribir, [1]:

$$\boldsymbol{Z}_k = \sum_{n=0}^{k-1} \boldsymbol{X}^n \boldsymbol{Y}, k \geq 1. \tag{9}$$

En la expresión (9) se observa que $\boldsymbol{Z}_k$ se puede obtener de forma recursiva como sigue:

$$\boldsymbol{Z}_k = \boldsymbol{Y} + \boldsymbol{X}\boldsymbol{Z}_{k-1} \tag{10}$$

A partir de la expresión (10), se obtiene en la Fig. 1 una arquitectura simple para el cálculo de $\boldsymbol{Z}_k$:

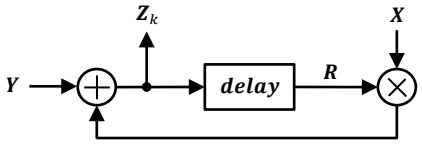

Fig. 1. Diagrama de bloques para la inversión de matrices [1].

Esta arquitectura y la propuesta en [1] son similares, ya que usan la misma expresión. En [1], la matriz de salida se obtiene invirtiendo los elementos de la diagonal en matrices de diagonal dominante. En este trabajo, la matriz de salida se calcula usando la inversa de la matriz adyacente anterior, que está correlada en tiempo o frecuencia con la matriz actual.

El sistema inversor de matrices completo sería:

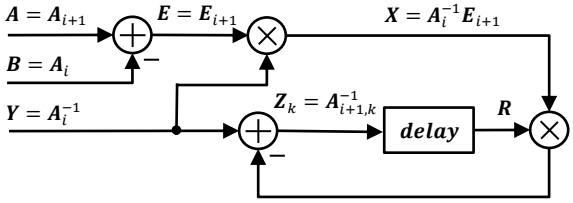

Fig. 2. Arquitectura completa para la inversión de matrices.

En el diagrama de la Fig. 2 se pueden observar dos partes diferenciadas. La primera encargada de calcular las matrices $\boldsymbol{X}$ e $\boldsymbol{Y}$ y la segunda del cálculo iterativo de la matriz inversa final $\boldsymbol{Z}_k$ usando las dos matrices anteriores.

Para obtener la matriz inversa $\boldsymbol{Y}$ se propone realizar una inversión inicial muy precisa con series de Neumann y la condición de diagonal dominante [1], durante un número de iteraciones suficiente, y luego, calcular el resto de inversas con el método presentado en este trabajo. No obstante, si el método de diagonal dominante no es factible, se puede obtener con uno de los métodos presentados en la sección I.

Un aspecto a considerar es que el error cometido se va acumulando, por lo que periódicamente hay que calcular la inversa de una forma más exacta.

IV. Implementación hardware

En esta sección, se describe cómo se ha implementado la arquitectura propuesta empleando una FPGA Xilinx Virtex-7 (XC7VX485T) [16], usando como lenguaje de descripción hardware VHDL y como herramienta de desarrollo ISE Design Suite 14.6.

A. Bloque multiplicador

En la arquitectura planteada, la operación más compleja a realizar es el producto matricial, ya que no es conmutativo. El bloque que realiza el producto, de ahora en adelante MMC (Multiplicador de Matrices Complejas), es el elemento base y de su arquitectura dependerá el diseño del resto del sistema.

Para su desarrollo se emplea el producto de matrices por bloques de submatrices. Sea una matriz $\boldsymbol{\Delta}$ de tamaño $(m\ x\ n)$ con p particiones en las filas y s particiones en las columnas y una matriz $\boldsymbol{\Phi}$ de tamaño $(n\ x\ q)$ con s particiones en las filas y r particiones en las columnas. El producto $\boldsymbol{\Omega} = \boldsymbol{\Delta}\boldsymbol{\Phi}$ se puede

obtener por bloques, donde $\boldsymbol{\Omega}$ es una matriz de tamaño $(m\ x\ q)$ con p particiones en las filas y r particiones en las columnas. Cada submatriz de $\boldsymbol{\Omega}$ se obtiene como sigue [17]:

$$\boldsymbol{\Omega}_{\alpha\beta} = \sum_{\gamma=1}^{s} \boldsymbol{\Delta}_{\alpha\gamma}\, \boldsymbol{\Phi}_{\gamma\beta} \tag{11}$$

En primer lugar, se desarrolla un bloque que realiza el producto de dos matrices de tamaño $4x4$, utilizando cuatro multiplicadores complejos y tres sumadores de dos entradas. Empleando la propiedad anterior, se desarrollan dos bloques, uno que multiplica matrices de tamaño $8x8$ y otro de $16x16$. Por lo que el tamaño de las matrices a invertir será $M \leq 16$, aunque el diseño es escalable a tamaños mayores. El bloque MMC realiza en cada ciclo de reloj la multiplicación de una fila de la primera matriz por una columna de la segunda, ambas de 16 elementos y cada elemento de 16 bits.

B. Diagrama de bloques del diseño en hardware

En la arquitectura propuesta en la Fig. 1 y la Fig. 2, el sistema trabaja a tasa matriz. Sin embargo, tras diseñar el bloque MMC, resulta necesario adaptar el diseño para que se trabaje a tasa fila/columna. De este modo cada muestra será un vector correspondiente a una fila/columna completa. En la Fig. 3 se representa el diagrama de bloques con esta modificación. La arquitectura consta de dos partes que están separadas con una línea discontinua. La primera se encarga del cálculo de $\boldsymbol{X}$ y la segunda es la parte iterativa, que una vez obtenida $\boldsymbol{X}$, calcula el resultado $\boldsymbol{Z}_k$. Para este diseño se supone que la matriz inversa $\boldsymbol{Y}$ es conocida de antemano.

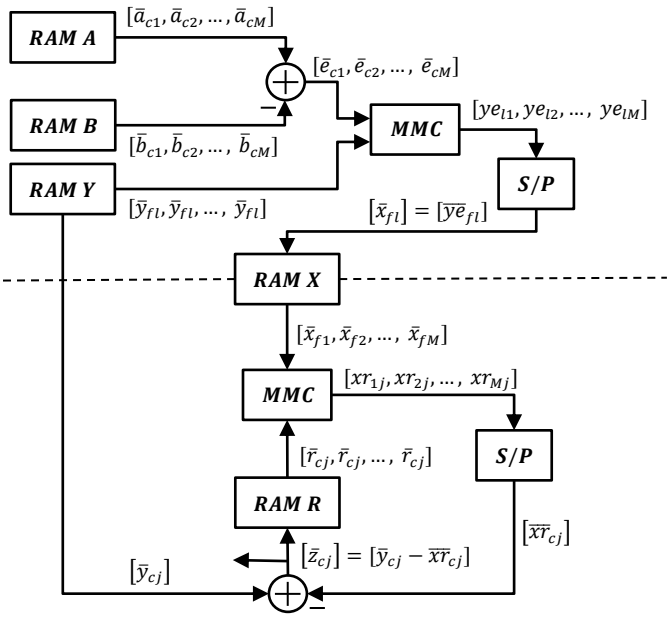

Fig. 3. Diagrama de bloques propuesto para la implementación hardware.

Los elementos de las matrices de entrada al sistema $\boldsymbol{A}$, $\boldsymbol{B}$ e $\boldsymbol{Y}$ son números complejos de 16 bits, tanto la parte real como la imaginaria. En las matrices $\boldsymbol{A}$ y $\boldsymbol{B}$ se utiliza un bit de signo, otro de parte entera y catorce bits para la parte fraccionaria. En cambio, en la matriz $\boldsymbol{Y}$ se utiliza un bit de signo y los quince bits restantes para la parte fraccionaria.

En el diagrama de la Fig. 3 se incluyen cinco bancos de memorias para almacenar las tres matrices de entrada $\boldsymbol{A}$, $\boldsymbol{B}$ e $\boldsymbol{Y}$ y las matrices intermedias $\boldsymbol{X}$ y $\boldsymbol{R}$. Cada banco consta de 16 memorias RAM de 16 posiciones. En cada RAM se almacena una de las 16 filas/columnas y en cada posición uno de los 16 elementos de cada fila/columna (32 bits por posición para almacenar parte real e imaginaria). Cada banco de memorias tiene asociado dos procesos encargados de la escritura y lectura de los datos en el orden deseado. Cuando $M < 16$ las posiciones sobrantes contendrán ceros, de forma que el bloque adapta el número de lecturas, reduciendo el número de operaciones a realizar a M^2, en lugar de 256.

El bloque que realiza la suma se desarrolla de forma que sume los elementos necesarios para alimentar al componente MMC. Esto es un bloque que sume dos filas/comunas, o dicho de otro modo, dos vectores de 16 elementos. Para ello se emplean 32 sumadores reales. Tanto el sumador como el bloque MMC realizan un recorte de aritmética a su salida a 16 bits con un bit de signo y 15 para la parte fraccionaria.

A la salida del bloque MMC se incluye un conversor de datos serie-paralelo, ya que el primero devuelve elementos, no vectores fila/columna que es lo que se necesita.

Una vez descritos los elementos principales, se va a pasar a analizar la parte superior del diagrama de bloques de la Fig. 3, que representa el cálculo de la fila l de la matriz $\boldsymbol{X}$. Cada vector que aparece entre corchetes corresponde a un instante temporal (un ciclo de reloj). Además, el subíndice c denota que se trata de una columna y el subíndice f de una fila. Las memorias RAM A y RAM B van leyendo cada columna, en cambio la memoria RAM Y lee siempre la fila l. Así, a la salida del bloque MMC se obtienen de uno en uno los elementos de la fila l de $\boldsymbol{X}$. Después, el conversor serie-paralelo se encarga de obtener la fila completa.

Una vez se ha obtenido $\boldsymbol{X}$, su valor se puede usar en la parte iterativa del circuito. En la parte inferior de la Fig. 3, debajo de la línea discontinua, se representa el cálculo de la columna j de la matriz inversa $\boldsymbol{Z}$ en cada iteración k. Inicialmente, la memoria RAM R debe estar cargada con la matriz $\boldsymbol{Y}$. En primer lugar, se va haciendo el producto de cada fila $\boldsymbol{X}$ por la misma columna j de $\boldsymbol{R}$. Así, se obtiene a la salida del conversor serie-paralelo la columna j de $\boldsymbol{XR}$. Este valor se resta a la columna correspondiente de la matriz $\boldsymbol{Y}$, obteniendo la columna j de la matriz $\boldsymbol{Z}$. Este valor se escribe en la RAM R, excepto las últimas M columnas de la última iteración que corresponden a la matriz inversa de salida.

V. Resultados y Prestaciones

En esta sección, se realiza en primer lugar un análisis del coste computacional del algoritmo propuesto. Además, se presentan los resultados obtenidos y prestaciones del sistema.

A. Análisis del coste computacional

Este apartado analiza la complejidad del algoritmo, en cuanto a número de productos y divisiones, ya que son las operaciones más costosas. Además, se compara con otros métodos de inversión, como los basados en descomposición QR y los basados en series de Neumann con condición de diagonal dominante. En la tabla I se muestran los resultados. Ahí se observa que la inversión basada en la descomposición QR es la más compleja. Aunque el método es más exacto, se requieren un gran número de productos y divisiones.

En comparación con el método de diagonal dominante, el número de multiplicaciones del algoritmo propuesto es un poco mayor, pero del mismo orden. Sin embargo, la técnica de diagonal dominante requiere realizar divisiones, lo que es una desventaja, porque es complicado implementar bloques

divisores. Otra desventaja de esta técnica es que hay situaciones en las que no se puede emplear, por ejemplo cuando el número de antenas en la *BS* no es suficiente.

TABLA I
ANÁLISIS DE COMPLEJIDAD

Método	Multiplicaciones	Divisiones
QR [7]	$\frac{28}{3}M^3+\frac{7}{2}M^2+\frac{7}{6}M$	$\frac{5}{2}M^2-\frac{3}{2}M$
Diagonal	$(k-2)(2M^3+3M^2+M)$ $+2(M^2-M)$ si $k>1$; en otro caso 0.	M
Propuesto	$2(k+1)M^3+2(k-1)M^2$ si $k>1$; en otro caso 0.	0

B. Evaluación de resultados y prestaciones

Este apartado se dedica a presentar los resultados y prestaciones del diseño implementado, tanto los recursos hardware ocupados como la latencia y el throughput.

En cuanto a latencia del sistema, la primera parte, que realiza el cálculo de $\boldsymbol{X}$, emplea M^2+15 ciclos. La parte iterativa tarda $(k-1)(M^2+14)-1$ ciclos en calcular $\boldsymbol{Z}_k$. La latencia del sistema completo es de $k(M^2+14)$ ciclos.

Al sintetizar el diseño, se obtiene que el sistema es capaz de trabajar con un periodo de reloj mínimo de 3.05 ns, lo que equivale a una frecuencia máxima de 327.6 MHz. Con la frecuencia del reloj y la latencia (en ciclos) se puede obtener tanto la latencia en segundos como el throughput (número de inversiones por segundo). En la tabla II se comparan estas métricas para valores típicos de frecuencia de reloj en LTE, distintos números de usuarios M y para $k=3$ iteraciones:

TABLA II
COMPARATIVA DE RENDIMIENTO

		Latencia (µs)		Throughput (MInv/s)	
Frec. reloj (MHz)		122.88	245.76	122.88	245.76
M	5	0.95	0.48	1.05	2.10
	10	2.78	1.39	0.36	0.72
	16	6.60	3.30	0.15	0.30

TABLA III
RECURSOS HARDWARE

Recurso FPGA	Utilización
FF slices	6776 (1.1%)
LUT slices	3303 (1.1%)
Block RAMs	40 (3.9%)
DSP48E1s	96 (3.4%)

En cuanto a recursos hardware, en la tabla III aparecen el número de recursos usados y el porcentaje de ocupación. Una forma de reducir los recursos empleados consiste en reutilizar componentes. Por ejemplo, se puede usar el mismo bloque MMC para las dos multiplicaciones, ya que ambos no se usan al mismo tiempo. Con esta mejora se reduce justo a la mitad el número de DSPs, casi a la mitad el número de FFs y prácticamente un tercio el número de LUTs. Para reducir el número de *Block RAMs* se podrían usar bloques con RAMs de 256 posiciones en lugar de con bancos de 16 RAMs de 16 posiciones. Esto supone una mayor la latencia del sistema, ya que así no se puede leer una fila/columna completa. Sólo se puede escribir y leer un elemento en el mismo ciclo de reloj.

VI. CONCLUSIONES

Se ha propuesto un método eficiente para la inversión de matrices en escenarios en los que existe una correlación en tiempo o frecuencia. En estas situaciones no es necesaria la condición de matrices con diagonal dominante, sino que se puede invertir cualquier matriz usando sólo productos y sumas siempre que la correlación entre las matrices sea alta. Además, se ha implementado un sistema capaz de invertir matrices de hasta $16x16$ con este método, usando una FPGA Virtex7 de Xilinx. El sistema, además de ser escalable para invertir matrices de tamaños superiores, presenta baja complejidad y un throughput alto, lo que es imprescindible en sistemas reales de comunicaciones móviles.

AGRADECIMIENTOS

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad, FEDER bajo la concesión TEC2016-80090-C2-1-R y por la Universidad de Málaga.

REFERENCIAS

[1] F. Wang, C. Zhang, J. Yang, X. Liang, X. You, and S. Xu, Efficient Matrix Inversion Architecture for Linear Detection in Massive MIMO Systems, IEEE ICDSP, Pp. 248-252, September 2015

[2] L. Fang, and D. Huang, Neumann Series Expansion Based LMMSE Channel Estimation for OFDM Systems, IEEE Communication Letters, Vol. 20, No.4, April 2016

[3] D. Zhu, B. Li, and P. Liang, On the Matrix Inversion Approximation Based on Neumann Series in Massive MIMO Systems, IEEE ICC 2015, Pp. 1763-1769, September 2015

[4] Z. Wu, C. Zhang, Y. Xue, S. Xu, and X.You, Efficient Architecture for Soft-Output Massive MIMO Detection with Gauss-Seidel Method, IEEE ISCAS 2016, Pp. 1886-1889, August 2016

[5] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, Large-Scale MIMO Detection for 3GPP LTE: Algorithms and FPGA Implementations, IEEE Journal of Selected Topics in Signal Processing, Vol. 8, No.5, October 2014

[6] X. Wang, Airvana and M. Leeser, A Truly Two-Dimensional Systolic Array FPGA Implementation of QR Decomposition, ACM Transaction on Embedded Computing System, Vol. 9, No.1 Article 3, October 2009

[7] L. Ma, K. Dickson, J. McAllister, J. McCanny, QR Decomposition-Based Matrix Inversion for High Performance Embedded MIMO Receivers, IEEE Tran. Signal Processing, Vol. 59, no. 4, April 2011

[8] Y. He, Y. Song, G. Du, D. Zhang, Research of Matrix Inversion Acceleration Method, CiSE 2009, pp. 1-4, December 2009

[9] K. Wang, L. Li, F. Han, H. Pan, F. Feng, X. Yu A High Performance Parallel VLSI Design of Matrix Inversion, IEEE 11th International Conference on ASIC (ASICON), pp. 1-4, July 2016

[10] D. Yang, G. D. Peterson, H. Li, J. Sun, An FPGA Implementation for Solving Least Square Problem, 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines, pp.303-306, Oct. 2009

[11] Y. Chen, H. Halbauer, M. Jeschke, R. Ritcher, An efficient Cholesky Decomposition based multiuser MIMO detection algorithm, IEEE PIMRC 2010, pp. 499-503, December 2010

[12] X. Gao, L. Dai, J. Zhang, S. Han, C. I Capacity-Approaching Linear Precoding with Low-Complexity for Large-Scale MIMO Systems, IEEE ICC 2015, pp. 1577- 1582, September 2015

[13] W. Song, X. Chen, L. Wang, X. Lu, Joint conjugate gradient and Jacobi iteration based low complexity precoding for massive MIMO systems, 2016 IEEE ICCC 2016, pp. 1-5, October 2016

[14] B. Yin, M. Wu, J. R. Cavallaro, C. Studer, VLSI design of largescale soft-output MIMO detection using conjugate gradients, IEEE ISCAS 2015, pp. 1498- 1501, July 2015

[15] Y. Jiang, M. K. Varanasi, J. Li, Performance Analysis of ZF and MMSE Equalizers for MIMO Systems: An In-Depth Study of the High SNR Regime, IEEE Transactions on Information Theory, Vol. 57, no. 4, pp. 2008-2026, March 2011

[16] Xilinx Inc., VC707 Evaluation Board for the Virtex-7 FPGA User Guide, 2016. [online]. Disponible en: www.xilinx.com.

[17] Harville, David A., Matrix Algebra From a Statistician's Perspective, Springer, 1997.