

Low Power Implementation for Minimum Norm Sorting and Block Upper Tri-angularization of Matrices used in MIMO Wireless Systems

Zahid Khan, Tughrul Arslan, John S. Thompson, Ahmet T. Erdogan
School of Engineering and Electronics,
The University of Edinburgh,
Mayfield Road, Edinburgh, EH9 3JL, Scotland, UK
z.khan@ed.ac.uk

Abstract

Multiple Input - Multiple Output (MIMO) wireless technology involves highly complex vectors and matrix computations which are directly related to increased power and area consumption. This paper proposes an area and power efficient VLSI architecture that can serve the dual purpose of minimum norm sorting of rows as well as upper/lower block tri-angularization of matrices. The resources inside the architecture are shared among both operations and only primitive computations are used. Results indicate saving in silicon real estate as well as power consumption compared to previous architecture without degrading performance.

1. Introduction

Multiple Input - Multiple Output (MIMO) wireless communication promises to remove the limits of wireless networks by providing spectral efficiency near Shannon's bound [1]. Because of its benefits, MIMO is entering into almost every wireless standard such as 802.11n for Wi-Fi and 802.16 for WiMax application. MIMO involves complex signal processing which is directly related to high power consumption and more cost in silicon [2][3].

VBLAST is a MIMO detection algorithm [4] that provides a good trade-off between BER (bit error rate) performance and computational complexity compared to its counter parts. Zero Forcing (ZF) and Minimum Mean Square Error (MMSE) detectors [5] are computationally less expensive than VBLAST; however, they provide inferior BER performance compared to VBLAST. The optimal solution, maximum likelihood (ML) [5] detection, provides best BER performance. However, it is highly expensive regarding computational complexity. This increases exponentially with the number of antennas used and is prohibitively high for antennas more than 4. Therefore, ML algorithm cannot be implemented on mobile platforms due to high overhead of area and power [3].

In VBLAST itself, the bottlenecks are repeated pseudo inverse, sorting and nulling vector calculation. This

repeated computation is a power hungry process. It also leads to numerical instability in hardware implementation which can be reduced using alternative algorithms such as the square root algorithm [6] to compute the pseudo inverse and sorting of the rows of the inverted channel matrix. Even with the square root algorithm [6], it is necessary to optimize the design for power and area.

In CMOS, sources of power consumption include short circuits, leakage currents and switching. The switching or dynamic power is described as

$$P = kC_L V^2 f$$

where k represents the switching activity factor, C_L the total physical capacitance, V the supply voltage and f the frequency of operation. Algorithmic power optimization includes reduction of both physical capacitance and switching activity factor. Physical capacitance can be reduced by reducing the area of hardware through efficient implementation [7]. Switching activity reduction either comes from area reduction that reduces the number of nodes or from reducing the switching frequency of nodes. One of the algorithmic optimizations is reducing redundancy [8] from a design. By reducing the redundant operations or hardware, unnecessary switching of the clock as well as other signals can be avoided for power saving.

This research work presents a novel VLSI architecture that performs minimum norm sorting and block upper tri-angularization of matrices by employing a series of unitary transformations known as Jacobi rotation [9]. The architecture is also compared with the only available architecture in the literature [10], where CORDIC algorithm is used for Jacobi transformation.

2. MIMO System Model and Square Root Algorithm for VBLAST

In MIMO communication systems, more than one antenna is used to transmit symbols and more than one antenna is used to receive them. In the diagram of Figure 1, spatial multiplexing is used in which M transmit antennas transmit M different symbols simultaneously while each symbol is received by the N receive antennas. Each symbol

transmitted is received by all the receiving antennas thus making multiple channel paths. These paths, if combined, make a matrix of NxM channel elements.

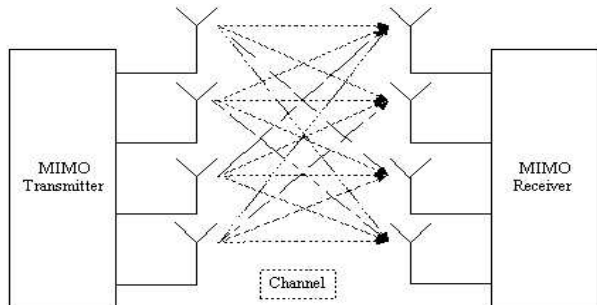


Figure 1:(MIMO System Model)

If $\mathbf{s} = [s_1, s_2, s_3, s_4, \dots, s_M]^T$ denotes the symbol vector transmitted, \mathbf{H} denotes the NxM channel matrix between the receive and transmit antenna array, and \mathbf{v} denotes the AWGN noise vector, then the corresponding received vector \mathbf{r} is given by

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{v} \quad (1)$$

To recover the transmitted symbol vector \mathbf{s} , it is necessary to invert the channel matrix. The inversion can be done depending upon the detection method. For MMSE, the channel matrix is augmented by the noise variance (α) and the detector output is (2)

$$\mathbf{s} = (\alpha\mathbf{I} + \mathbf{H}^*\mathbf{H})^{-1} \mathbf{H}^* \mathbf{r} \quad (2)$$

where * represents complex conjugate transpose.

In VBLAST, successive nulling and cancellation is used to detect the transmitted symbols. The channel matrix is first inverted and then sorted to detect that symbol first which has the highest post detection Signal to Noise ratio (SNR). This corresponds to the row of the inverted channel matrix having minimum norm distance. The detected symbol is subtracted from the received symbol vector. The corresponding column of the \mathbf{H} matrix is zeroed down and the process is repeated with the deflated channel matrix until all the symbols are detected.

In VBLAST detection, square root algorithm [6] is used for pseudo inversion and minimum norm sorting. The square root algorithm computes \mathbf{QR} decomposition of the augmented channel matrix given by (3) in a series of unitary transformations.

$$\begin{bmatrix} \mathbf{H}^{NxM} \\ \sqrt{\alpha}\mathbf{I}^{NxM} \end{bmatrix} = \mathbf{QR} = \begin{bmatrix} \mathbf{Q}_a^{NxM} \\ \mathbf{x} \end{bmatrix} \mathbf{R}^{MxM} \quad (3)$$

The algorithm first decomposes the channel matrix into \mathbf{QR} and then computes $\mathbf{P}^{1/2} = \mathbf{R}^{-1}$ and $\mathbf{B}_N = \mathbf{Q}_a$ from which

pseudo inverse $\mathbf{P}^{1/2} \mathbf{Q}_a^*$ can be computed. The sorting part of the algorithm is explained below [6]:

1. Find the minimum length row of $\mathbf{P}^{1/2}$ and permute it to be the last (Mth) row. Permute \mathbf{s} accordingly.
2. Find a unitary Σ such that $\mathbf{P}^{1/2} \Sigma$ is block upper triangular:

$$\mathbf{P}^{1/2} \Sigma = \begin{bmatrix} \mathbf{P}^{(M-1)/2} & \mathbf{P}_M^{(M-1)/2} \\ 0 & P_M^{1/2} \end{bmatrix}$$

3. Update \mathbf{Q}_a to $\mathbf{Q}_a \Sigma$.

3. Proposed Pipelined VLSI Architecture

The block diagram of the novel pipelined architecture is shown in Figure 2. This architecture serves two purposes

1. Minimum Norm Sorting
2. Block Upper tri-angularization of square matrices

The proposed architecture is designed to share resources between the two processes in a time multiplexed fashion. The architecture first sorts rows of $\mathbf{P}^{1/2}$ for minimum norm and then uses Jacobi transformation to make all but the last element of that row zero. Before sorting to start, $\mathbf{P}^{1/2}$ and \mathbf{Q}_a are taken from the pseudo inverse and stored in the two dual port rams (duram1 and duram2) as shown in Figure 3.

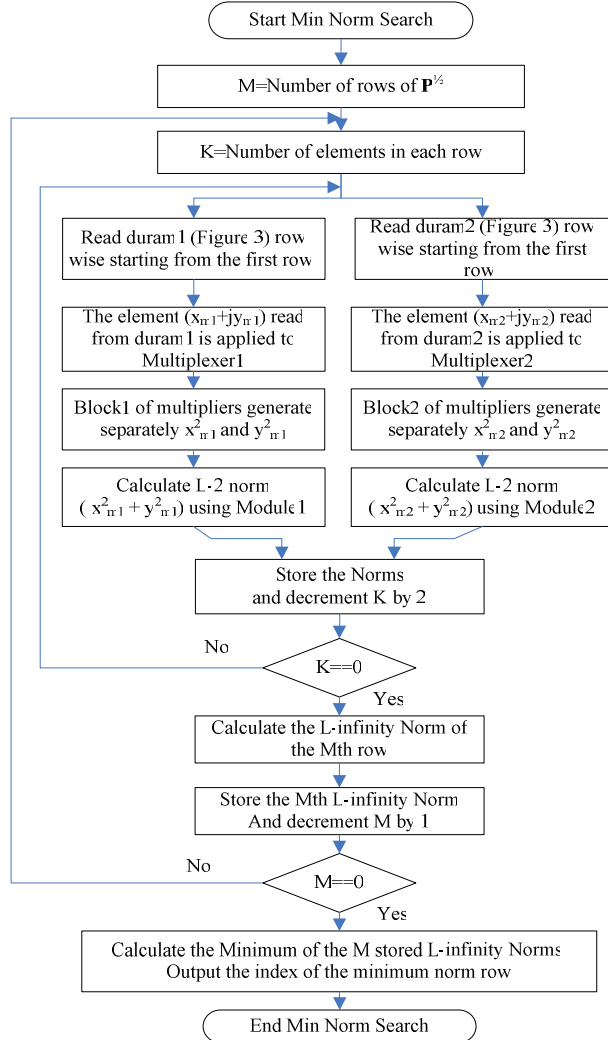
3.1 Minimum Norm Sorting

The architecture first computes the L-infinity norm of all the M rows of $\mathbf{P}^{1/2}$ using the equation given below:

$$\|\mathbf{x}\|_{\infty} = \max_i(|x_i|) \quad [9] \quad (4)$$

For minimum norm sorting, we have developed an algorithm which is described by flow graph in Figure 4.

1. Read duram1 and duram2 row wise starting from the first row. During each read cycle, two elements are read, one from duram1 and the other from duram2.
2. The element $(x_{m1} + jy_{m1})$ read from duram1 is applied to Multiplexer1 while that $(x_{m2} + jy_{m2})$ read from duram2 is applied to Multiplexer2.
3. The outputs from block1 $(x_{m1}^2$ and $y_{m1}^2)$ are added in Module1. Similarly the outputs from block2 $(x_{m2}^2$ and $y_{m2}^2)$ are added in Module2. It should be noted that the L-2 norm of a complex number $x + jy$ is $\sqrt{x^2 + y^2}$ and the L-infinity norm of a row is the maximum of L-2 norms of the individual elements.



(M=K=4 is assumed here)

Figure 4: (Flow graph of the minimum norm search)

However, in this application, the L-infinity norm of the row can be calculated without taking the square root as is normally done in L-2 norms of the individual elements. This implies that square of the L-2 norms will be compared. This will not effect the overall result for the reason that if $k > n$, then $\sqrt{k} > \sqrt{n}$. By not taking the square root, we are saving more silicon real estate as well as power consumption.

4. Store the two norms in the L-infinity norm module
5. Repeat steps 1 to 4 for the other two elements of the same row and store them in the L-infinity norm module.
6. Calculate the L-infinity norm which is the maximum of the four norms for the four individual elements and pass it to the min norm search module
7. Repeat steps 1 to 6 for other rows of $\mathbf{P}^{1/2}$ and apply their L-infinity norms to the min norm module.
8. The min norm module finds the minimum of the four norms and remembers the index of the row for which

the norm is the least. The index is applied to the control module for read/write address and other control signal generation for zeroing all but the last element of the indexed row.

3.2 Block upper tri-angularization

This section describes zeroing the elements of the minimum length row using Jacobi transformation. Since the architecture uses a number of multipliers and a divider, the following algorithm is used to perform the Jacobi transformation.

Given $\mathbf{x} \in \mathbf{R}^n$ [9] and indices i and k that satisfy $1 \leq i < k \leq n$, the following algorithm computes $c = \cos(\theta)$ and $s = \sin(\theta)$ such that the k -th component of $J(i, k, \theta)x$ is zero.

$$\text{If } x_k = 0 \text{ then } c = 1 \text{ and } s = 0 \quad (5)$$

$$\text{else if } |x_k| \geq |x_i| \text{ then}$$

$$t := x_i / x_k, s := 1 / (1+t^2)^{1/2}, c := st \quad (6)$$

$$\text{else } t := x_k / x_i, c := 1 / (1+t^2)^{1/2}, s := ct \quad (7)$$

In channel estimation or pseudo inverse computation, it is highly unlikely that $t=1$, t will have a value less than 1. This implies that values of 's' or 'c' can be approximated using the Taylor's series given by

$$s := (1+t^2)^{-1/2} = 1 - 2^{-1}t^2 \quad (8)$$

This shows the values of 's' or 'c' can be computed using multipliers, shifters and adders. The contribution of the higher terms is insignificant and ignored.

The steps in tri-angularization are explained below:

Step 1:

The first step in tri-angularization is to convert the four elements in the minimum norm row of $\mathbf{P}^{1/2}$ matrix from complex to real by zeroing their imaginary parts [11]. In this step, the four complex elements in the minimum length row of both durams are converted to real. To convert them into real, we need 't' to calculate 'c' and 's' and then treat the real and imaginary parts as two real elements.

- a- Calculate 't' using divider for each of the complex element (say p_{11}, p_{12}, p_{13} and p_{14}).
- b- Use 't' to compute 's or c' by applying 't' to multipliers (block1) and adders in module1. Store 's or c'. Compute 'c or s' using 't' and previously stored 's' or 'c' values.
- c- Use the values of 'c' and 's' to rotate the imaginary parts of the corresponding channel elements to zero using block1 together with module1 for p_{11} and p_{13} and block2 together with module2 for p_{12} and p_{14} respectively.

A 't' value is computed by reading a complex element from memory and applying it to the divider, the output of which is the corresponding 't' value which is not only stored but also applied to the multipliers in block1 to compute either 's' or 'c'. For example, if 't' is calculated using equation 6 then 's' is computed using equation 8 by applying 't' to block1 to form 't²' which is then applied to module1 to compute '-2⁻¹t²'. The 's' value so obtained is used to compute 'c' value using equation $c = s*t$. The 's' and 'c' values are obtained for all four elements in pipeline.

The imaginary parts are rotated to zero by applying both the real and imaginary parts of the complex elements together with their corresponding 'c' and 's' values to the two blocks of multipliers and adder modules in Jacobi rotation. The Jacobi rotation affects the entire column, therefore, if we want to zero the imaginary part of p_{11} , the entire column is involved in rotation. The effect of this rotation on p_{11} is to zero its imaginary part while the effect of the same rotation on all other elements of the same column is to change the phase angle by the phase angle of p_{11} . Since it is inherent in Jacobi rotation, all other columns will be modified in a similar manner. (For more explanation, the reader is referred to literature on Jacobi rotation.)

Step 2:

Rotate the second and third elements in the minimum length row of $\mathbf{P}^{1/2}$ to zero. (p_{11} , p_{12} , p_{13} , and p_{14} used in this section carry different values from step 2 and they are all real).

- a- Calculate 't' for the pair of p_{11} , p_{12} , and for p_{13} , p_{14}
- b- Compute 'c' and 's' for the corresponding 't'
- c- Rotate p_{12} and p_{13} to zero

For this, p_{11} is rotated against p_{12} while p_{13} is rotated against p_{14} . The calculation of 't', 'c' and 's' are as explained above. These 's' and 'c' values are then applied to the two blocks of 4 multipliers to rotate the entire column of p_{11} against the entire column of p_{12} . The eight multipliers are needed for the rotation of two complex numbers. This can be explained by assuming $h_{11}=r_1+jim_1$ and $h_{12}=r_2+jim_2$, then

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} r_1 + jim_1 \\ r_2 + jim_2 \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + j \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} im_1 \\ im_2 \end{bmatrix} \quad (10)$$

From the above expression, it is clear that 8 multipliers are used to compute real and imaginary values of the two columns that are involved in rotation. Though p_{11} and p_{12} are real but all other elements in the same columns have real and imaginary parts. After rotating p_{12} to zero, p_{13} and p_{14} are applied which rotate p_{13} to zero.

Step 3:

Rotate the first element p_{11} to zero

- a- Calculate 't', 'c', and 's' values for p_{11} and p_{14}
- b- Rotate p_{11} to zero

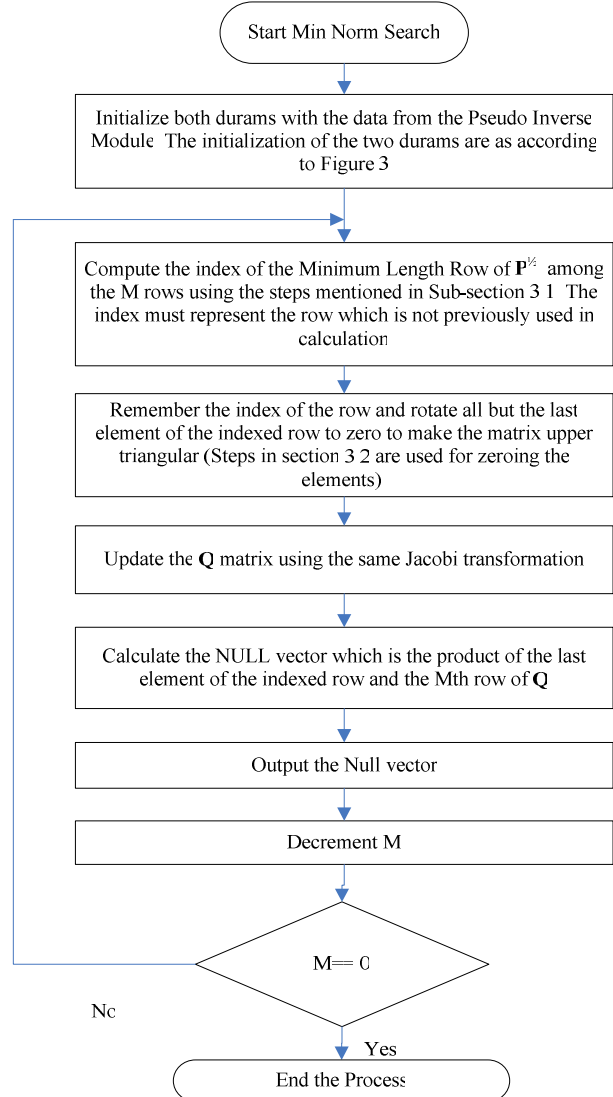


Figure 5: (Flow Graph for the sequence of operation)

The values 't', 'c' and 's' are calculated in a similar way as explained in step 1 and using 'c' and 's' values p_{11} is rotated to zero against p_{14} . At the end of step 3, all but the last element of the indexed row are zeroed.

3.3 Minimum Sort Algorithm Implementation

The sequence of operation in implementing the sorting part of the square root algorithm as described in section 2 is presented in the flow graph as shown in Figure 5.

The two durams (duram1 and duram2) are initialized as shown in Figure 3. The index of the minimum norm row is computed using the steps in section 3.1. The index is applied to the control unit which generates the necessary read/write addresses and other signals necessary for zeroing all but the last element of the indexed row. For zeroing the elements, steps explained in section 3.2 have been used.

There is no need to permute the minimum norm row to be the last row as only the indexes of the rows are permuted. The NULL vector is obtained as the product of the last element of the indexed row with the complex conjugate transpose of the Mth row of \mathbf{Q} . After calculating the first NULL vector, the Mth row of \mathbf{Q} and $\mathbf{P}^{1/2}$ are discarded. The process is repeated with M-1 rows for $\mathbf{P}^{1/2}$ and \mathbf{Q} . For the second NULL vector calculation, the remaining (M-1) rows of $\mathbf{P}^{1/2}$ are searched to find the next minimum norm row. The index of that row is applied to the control section in a similar way that is done for the first index. Proper control signals are generated to zero all but the last element of the indexed row. The second NULL vector is obtained by multiplying the last element of the indexed row with the (M-1)th row of \mathbf{Q} . The process is repeated until all M NULL vectors are calculated.

4. Simulation and Synthesis results

The architecture has been synthesized using Synopsys Design Compiler and mapped to *0.18um* CMOS technology. The area breakdown of the proposed module is given in Table 1. The control logic is taking about 27% of the entire area. This is due to the use of appreciable number of registers for read and write address generators as well as for storing values of 't', 'c', and 's'. Flipflop based dual port rams take 27.7% of the area which can be reduced by using latch based ram. The rest are taken by 8 multipliers and a divider. The CORDIC based SORTING module has not been synthesized. However, from the literature [12], the area of the pipelined CORDIC can be obtained. If SORTING module is developed based on CORDIC, there will be three CORDICs with one (called θ CORDIC) used for angle calculation while the other two (called Φ CORDICs) will be used for rotation. The area of the three CORDICs from [12] is about $767960 \mu\text{m}^2$. There will also be control unit for sequencing the operation of these modules. Therefore, the combined area will be much more than the area occupied by multiplier based SORTING module given in Table 1.

The maximum operating frequency of our synthesized module is 50MHz due to the combinational divider module. However, with a two-stage pipelined divider module, the maximum frequency of operation can be increased to 100MHz. The proposed module is simulated at 40MHz and the power figures of individual modules are given in Table 2. The divider is consuming 17.8% of power which can be reduced since the divider is active only during 5% of total processing time. During the time the divider is not active, its inputs can be gated to bar it from switching. The use of latch based ram can also result in further power reduction.

From [12], it is evident that the CORDIC based module consumes more power compared to the proposed module if simulated at the same clock frequency. The only advantage of the CORDIC based module is that its frequency of operation is above 100MHz while with the proposed

architecture a 100MHz can be obtained if a two stage pipelined divider is used.

Further power reduction with our module can be achieved by gating the clock to those modules that remain inactive for some period during the entire execution time of the sorting operation.

5. Conclusion

The authors have presented an area and power efficient pipelined VLSI architecture that performs the dual function of both sorting as well as block upper triangularization of matrices used in MIMO wireless systems. The architecture is based on primitive computational blocks and more area and power efficient compared to a CORDIC based architecture. The architecture exploits the parallelism inherent in Jacobi rotation.

6. References

- [1] G. J. Foschini, "Layered Space-Time architecture for wireless communication in fading environments when using multiple antennas," Bell Labs Tech. J., vol 2, Autumn 1996
- [2] G. Lawton, "Is MIMO the future of wireless communications?", computer, vol: 37, issue 7, July 2004 Pages 20-22
- [3] D. Garrett, L. Davis, St. Brink, B. Hochwald, G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding", Solid-State Circuits, IEEE Journal, vol. 39, Issue 9, Sept. 2004, Pp(s):1544-52
- [4] P.W. Wolniansky, G.J. Foschini, G.D. Golden, and R.A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel", Proc. ISSSE'98, Sept. 1998
- [5] A. Adjoudani, E.C. Beck, A.P. Burg, G.M. Djuknic, T.G. Gvoth, D. Haessig, S. Manji, M.A. Milbrodt, M. Rupp, D. Samardzija, "Prototype experience for MIMO BLAST over third-generation wireless system", Selected Areas in Communications, IEEE Journal on Vol. 21, Issue 3, April 2003 Page(s):440 – 451
- [6] B. Hassibi, "An efficient square-root algorithm for BLAST", Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on, Volume 2, 5-9 June 2000 Page(s):II737 - II740 vol.2
- [7] M.Pedram, "Power Minimisation in IC Design: Principles and Applications", ACM Transactions on Design Automation of Electronic Systems, vol. 1, no. 1, pp. 3-56, January 1996.
- [8] X. Wu, M. Pedram, L. Wang, "Multi-code state assignment for low power design", Circuits, Devices and Systems, IEE Proceedings vol. 147, Issue 5, Oct. 2000 Page(s):271 - 275
- [9] Gene. H. Golub, Charless F. Van Loan, "Matrix Computation"
- [10] Z.Guo, P.Nilsson, "A VLSI implementation of MIMO detection for future wireless communications", Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on, vol. 3, 7-10 Sept. 2003 Pages:29-49
- [11] C.M. Rader, "VLSI systolic arrays for adaptive

nulling”, (radar) Signal Processing Magazine, IEEE , vol. 13 , Issue: 4 , July 1996, Pages:29 – 49
 [12] Z. Khan, T. Arslan, J. S. Thompson and A T. Erdogan, “Area & Power Efficient VLSI Architecture for Computing Pseudo Inverse of Channel Matrix in a MIMO Wireless

System”, 19th International Conference on VLSI Design (VLSI Design 2006), pp. 734-737, Hyderabad, India, January 3 - 7, 2006.

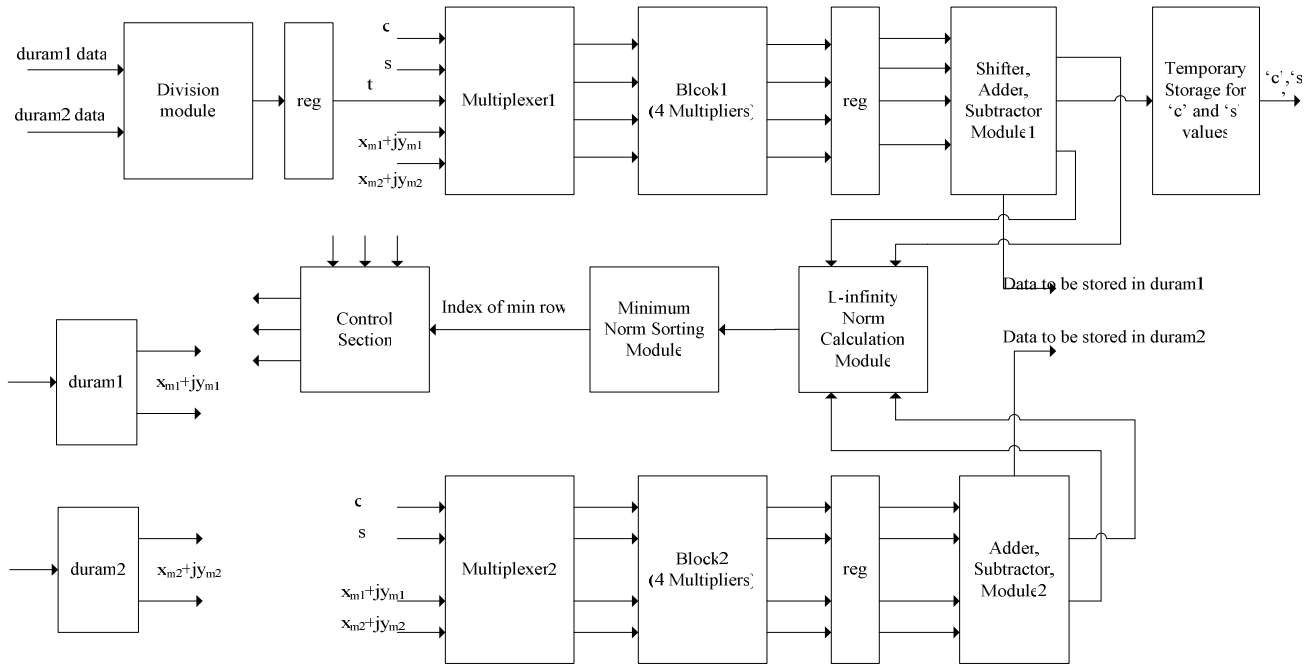


Figure 2: (Block diagram of Pipelined Architecture for Minimum Norm Sorting and Block upper tri-angularization of square matrices)

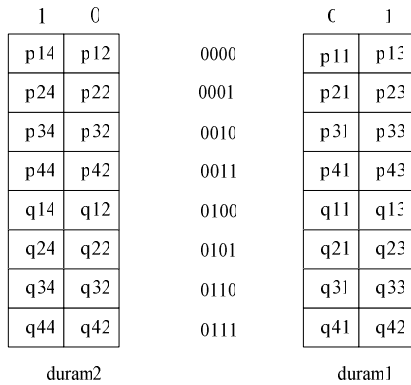


Figure 3: (Memory Management)

Table 1: Area breakdown of SORTING Module

	Quantity used	Area in μm^2	% Area distribution
Divider	1	60617	13.1
Multiplier	8	131784	28.6
Duram1	1	63765	13.9
Duram2	1	63765	13.9
Min Norm and L-infinity module	1	17871	3.9
Control Unit and glue logic		122445	26.6
Total		460248	100

Table 2: Power breakdown of SORTING Module

	Quantity used	Power in mw	% Power distribution
Divider	1	4.963	17.8
Multiplier	8	8.638	26.8
Duram1	1	2.834	10.2
Duram2	1	2.891	10.4
Min Norm & L-infinity module	1	0.671	2.4
Control Unit and glue logic		7.995	32.4
Total		27.922	100