

# A New Pipelined Implementation for Minimum Norm Sorting used in Square Root Algorithm for MIMO-VBLAST Systems

Zahid Khan, Tughrul Arslan, John S. Thompson, Ahmet T. Erdogan  
School of Engineering and Electronics,  
The University of Edinburgh, Mayfield Road, 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 and is prohibitively high for antennas more than 4. Therefore, the ML algorithm cannot be implemented on mobile platforms for its 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.

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 [7]. The architecture is also compared with the only available architecture in the literature [8], where CORDIC algorithm is used for Jacobi transformation.

The rest of the paper is organized such that section 2 describes MIMO system model and detection algorithm, section 3 proposes a novel architecture for minimum norm sorting and block upper tri-angularization, while section 4 presents results and section 5 concludes the paper.

## 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.

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 ( $\alpha$ ) 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.

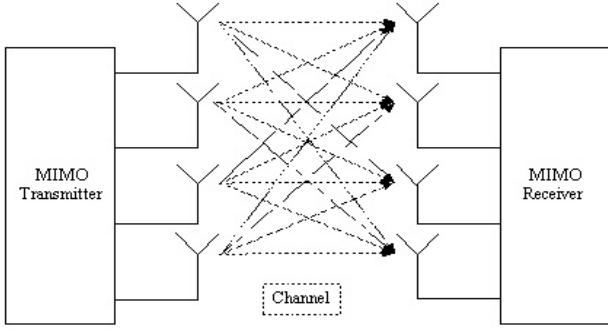


Figure 1: (MIMO System Model)

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  $\Sigma$  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  $\mathbf{Q}_a \Sigma$ .
4. The nulling vector for the M-th signal is given by  $P_M^{1/2} q_{\alpha, M}^*$ , where  $q_{\alpha, M}$  is the M-th row of  $\mathbf{Q}_a$
5. Go back to step 1 but now with  $\mathbf{P}^{(M-1)/2}$  and  $\mathbf{Q}_a^{(M-1)}$  (the first M-1 rows of  $\mathbf{Q}_a$ )

### 3. Proposed Pipelined VLSI Architecture

The block diagram of the novel pipelined VLSI architecture is shown in Figure 2 (last page). This architecture serves two purposes

1. Minimum Norm Sorting
2. Block Upper triangularization of square matrices

The proposed architecture is designed to share resources between these two processes in a time multiplexed fashion. The architecture first sorts rows of  $\mathbf{P}^{1/2}$  for minimum norm row 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 process and stored in the two dual port rams (duram1 and duram2) as shown in Figure 3 (last page)

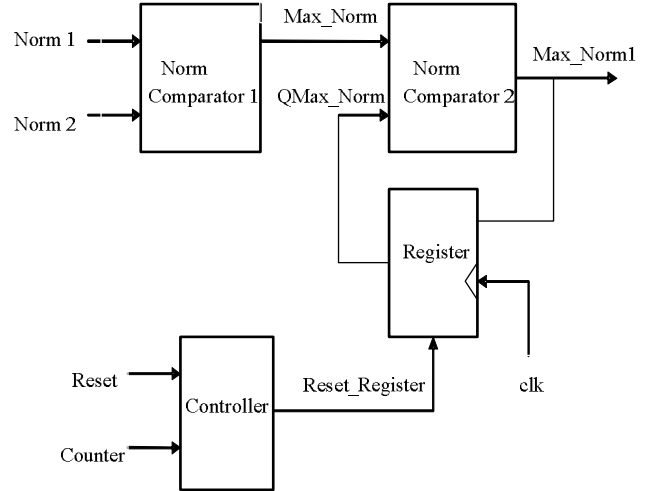


Figure 4: (L-Infinity Calculation Module)

#### 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:

$$\|\mathbf{x}\|_{\infty} = \max_i (|x_i|) \quad [10] \quad (4)$$

The L-Infinity norm module (shown in Figure 4) consists of two combinational comparators, register and a controller. Norm1 from block1 and Norm2 from block2 of Figure 2 are input to Norm Comparator1 which selects the maximum of the two values and assigns it to Max\_Norm. This value is taken by Norm Comparator2 which compares

it with the already stored value QMax\_Norm. Initially at the first clock cycle, QMax\_Norm is zero and, therefore, Max\_Norm1 is assigned the value of Max\_Norm. Max\_Norm1 is stored in the register. At the second clock cycle, the Norm Compartment1 provides the maximum of the remaining two values. This value is then compared with the previous maximum value stored in the Register and in two clock cycles the maximum of the four norms is calculated which is the required L-Infinity norm. Controller is used to reset the Register for the next L-Infinity norm calculation. The Controller takes two inputs, one is the global reset signal and the other counter that counts the number of cycles and its value is used for synchronizing different tasks and operations during sorting and triangularization. This circuit generates L-Infinity norm at every second clock cycle and therefore, takes 8 cycles to compute four norms.

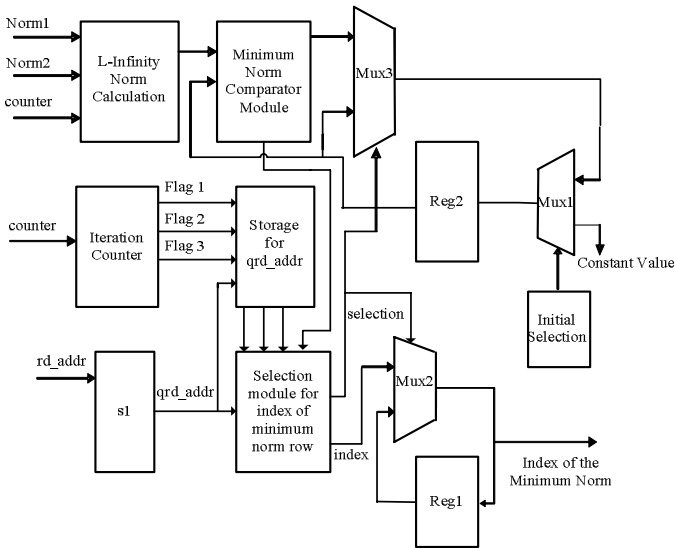


Figure 5: (Minimum Norm Search Module)

This L-Infinity module is used in Minimum Norm Search Module of Figure 5 for calculating the index of the minimum norm row. The iteration counter counts the number of iterations by taking the counter as input. A iteration is the duration in time during which the first nulling vector is calculated. It is expressed in the form of number of clock cycles. A specific value of the counter increments the iteration counter that is used to set one of the three flags named flag1, flag2 and flag3. Flag1, flag2 and flag3 are asserted only during iteration 1, 2 and 3 respectively. The flags are also de-asserted if a new process is started or global reset is asserted. The flags are used as enable inputs to control the storage of the indexes of the previous minimum rows searched by the module. This storage is necessary because in every iteration, the L-infinity norm module provides norms for all the four rows of  $\mathbf{P}^{1/2}$  regardless of the previously computed minimum norms.

This is done in order to simplify the hardware architecture for computing L-infinity norm.

The two least significant bits of the read address rd\_addr are used as index to the rows of  $\mathbf{P}^{1/2}$ . They are stored in s1. The s1 is basically two cascaded registers to produce a delay of two clock cycles in order to synchronize the index value with its corresponding L-infinity norm value. The 'Selection module for the index of minimum norm row' compares the current indexed value 'qrd\_addr' with the indexes stored in 'storage for qrd\_addr' and activates the select signal only if the qrd\_addr does not match with the already stored indexes. The 'Minimum Norm Comparator' compares the output of 'L-infinity norm' with the previous minimum value stored in 'Reg2'. 'Mux3' selects depending upon the 'select' signal either 'qmin\_norm' or 'min\_norm'. 'Mux1' selects either the output of Mux3 or a constant value to be stored in 'Reg2'. This value is initially stored at the start of every iteration and is kept in 'Reg2' for some time to be replaced by a valid minimum value from minimum norm search modules. 'Mux2' selects the correct value of the index corresponding to the minimum norm and stores it in 'Reg1'.

The sequence of operation is such that at the start of every iteration, Reg2 is loaded with the constant value and is kept in the register for a few clock cycles until it is replaced by minimum norm value coming through Mux3. L-infinity norm calculates the four norm values and pass them to 'Minimum Norm Comparator' module every other clk\_2. This clock is the half of the original system clock. The 'Minimum Norm Comparator' computes the minimum value. The index corresponding to the minimum norm row is selected by 'Mux2' taking decision inputs from the 'Minimum Norm Comparator' and index compare modules. The index thus selected is sent to the main controller for generating corresponding read, write and control signals.

### 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  $\mathbf{x} \in \mathbf{R}^n$  [7] 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.

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

$$\text{else if } |x_k| \geq |x_i| \quad \text{then} \\ t := x_i / x_k, s := \frac{1}{\sqrt{(1+t^2)^{1/2}}}, c := st \quad (6)$$

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

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 [9]. 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 's' which is then applied to module1 to compute ' $-2^{-1}t^2$ '. 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.

### 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

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 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 (see Figure 2). 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. Low Power strategies, Simulation and Synthesis results

Power reduction with our module can be achieved by gating the clock to those portions that remain inactive for some period during the entire execution time of the sorting operation. 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. All of these have been incorporated into the architecture.

The low power architecture has been synthesized using Synopsys Design Compiler and mapped to  $0.18\mu\text{m}$  CMOS technology. The area breakdown of the proposed module is given in Table 1. The control logic is taking about 29.7% 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'. Latch based dual port rams take 20% of the area. The rest are taken by 8 multipliers and a divider. The CORDIC based SORTING module has not been synthesized. However, from the literature [10], 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  $\theta$  CORDIC) used for angle calculation while the other two (called  $\Phi$  CORDICs) will be used for rotation. The area of the three CORDICs from [10] 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. From [10], 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.

#### 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] Gene. H. Golub, Charless F. Van Loan, "Matrix Computation"
- [8] 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
- [9] C.M. Rader, "VLSI systolic arrays for adaptive nulling", (radar) Signal Processing Magazine, IEEE , vol. 13 , Issue: 4 , July 1996, Pages:29 – 49
- [10] Zahid Khan, Tughrul Arslan, John S. Thompson and Ahmet T. Erdogan, "Enhanced Dual Strategy based VLSI Architecture for Computing Pseudo Inverse of Channel Matrix in a MIMO Wireless System", IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2006), Karlsruhe, Germany, March 2-3, 2006.

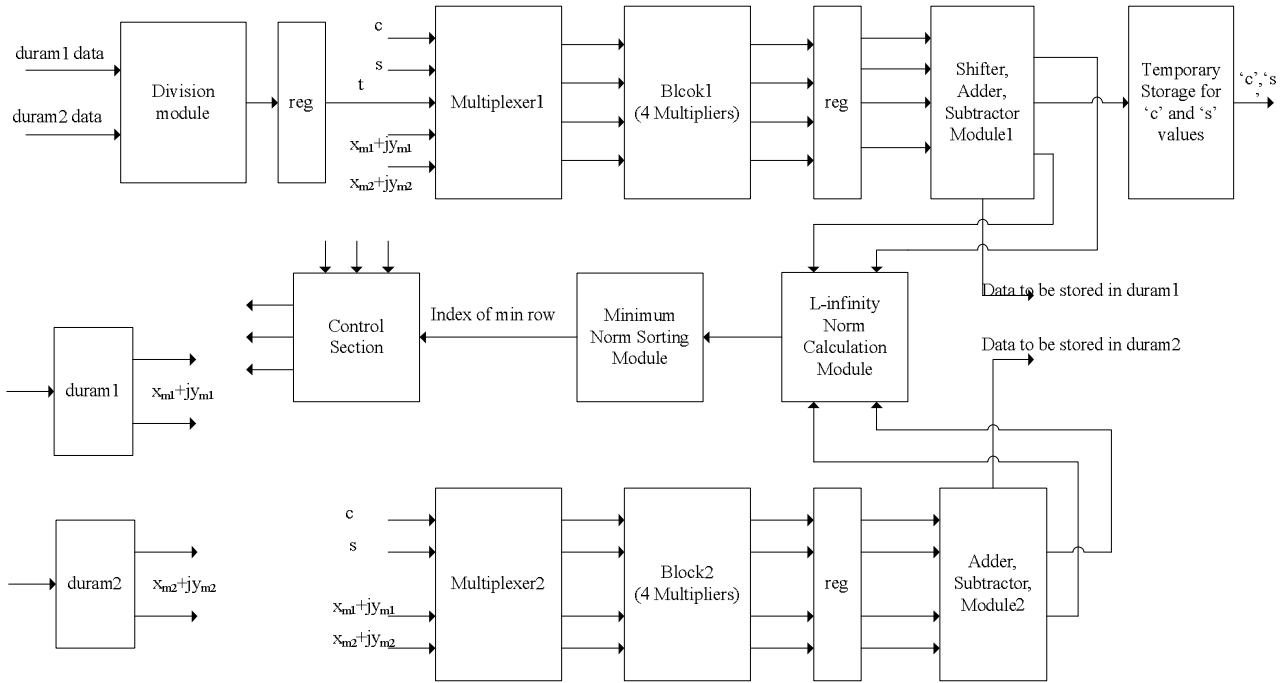


Figure 2: (Block diagram of Pipelined Architecture for Minimum Norm Sorting and Block upper tri-angularization of square matrices)

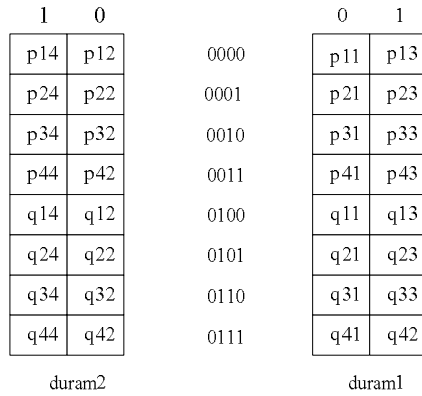


Figure 3: (Memory Management)

Table 1: Area breakdown of SORTING Module

	Quantity used	Area in $\mu\text{m}^2$	% Area distribution
Divider	1	60617	14.6
Multiplier	8	131784	31.8
Duram1	1	40633	9.8
Duram2	1	40832	9.8
Min Norm and L-infinity module	1	17871	4.3
Control Unit and glue logic		122445	29.7
<b>Total</b>		<b>414182</b>	<b>100</b>

Table 2: Power breakdown of SORTING Module

	Quantity used	Power in mw	% Power distribution
Divider	1	1.168	6.0
Multiplier	8	8.460	43.3
Duram1	1	1.085	5.6
Duram2	1	1.062	5.5
Min Norm & L-infinity module	1	0.671	3.5
Control Unit and glue logic		6.983	36.1
<b>Total</b>		<b>19.429</b>	<b>100</b>