High Throughput Iterative VLSI Architecture for Cholesky Factorization based Matrix Inversion

D. N. Sonawane[1] and M. S. Sutaone[2]
[1]Department of Instrumentation & Control
[2]Department of Electronics and Telecommunication
College of Engineering, Pune, India

ABSTRACT
Cholesky factorization is the computationally most expensive step in numerically solving positive definite systems. Due to inherently recursive computation process and associated floating point division and square root operations in Cholesky factorization, it is very difficult to obtain acceleration by exploiting parallelism on FPGA's. To solve this problem, approach suggests iterative architecture with parallelly fetching the matrix elements using customized Diagonal Processing Elements (DPU), Non Diagonal Processing Elements (NDPU) and Triangular Processing Elements (TPU) as computational processing units. The use of LNS approach using LUT technique for floating point square root and division arithmetic eventually improves resource and clock cycle utilization. Scheme is implemented using Xilinx Virtex-4 FPGA and achieves 0.032µs clock latency and obtained a throughput of 31.25Mupdates/s operating at 125 MHz for 4x4 matrix inversion problem.

Keywords
Choleskey Factorization, FPGA's, Iterative Architetcure, Virtex-4,

Nomencleture

FPGA	Field programmable Gate Array
DPU	Diagonal Processing Units
NDPU	Non-Diagonal Processing Units
TPU	Triangular Prcessing Units
PE's	Processing Elements

1. INTRODUCTION
A broad range of complex scientific applications requires Cholesky factorization, such as solving set of linear equations [1], the least square methods used in signal processing and image processing applications [4], getting inverse of positive definite Hessian matrix. In particular, obtaining inverse of Hessian matrix is a key computational task involved in interior point optimization method (IPM) and Active Set Method (ASM) used to solve quadratic programming (QP) problems. However, the computational cost to obtain inverse of Hessian matrix is heavy due to associated floating point square root and division operations. It is therefore very difficult to obtain hardware acceleration of Cholesky factorization. Variety of methods is available for implementing Cholesky factorization algorithm by exploiting the specific features of different computational systems.

Field Programmable Gate Arrays (FPGA's) are perfect platforms to implement arithmetic operations such as matrix multiplication, matrix inversion, factorization, square root, multiply/divide, etc. Due to their powerful architectural features like Block RAM, embedded multipliers, shift registers (LUTs), Digital Clock Manager (DCM), re-configurability and parallelism; FPGAs have become more powerful tool for algorithm prototyping and IP core development.

In the earlier studies [1~10] researchers have targeted to implement Cholesky factorization algorithm using different architectures and various FPGA platforms. Antonio Roldao andGeorge A. Constantinides [1] propose Cholesky Factorization based high throughput floating point Conjugate Gradient method architecture for dense matrices. Their design mainly targets the parallelism and deeply-pipelined architecture for CG method using Virtex5 FPGA, achieves a comparable performance compared to CPU Implementation. Depeng Yang et.al [3] [4] has compared the performance of Cholesky factorization implementation using FPGA and GPUs, authors has proposed highly parallel architecture and claims that FPGA implementation achieves higher clock cycle efficiency than GPU implementation. Cholesky factorization for solving least square problems and their FPGA implementation described by [5], author achieves good performance over general purpose CPU implementation by optimizing the algorithm using dedicated hardware architecture for solving triangular linear equations. Junqing Sun, et.al [2], introduces high-performance mixed-precision linear solver using FPGA's. In this paper author proposes the idea of utilizing lower performance floating point data format to achieve higher computational performance by introducing mixed-precision iterative refinement algorithms.

Here therefore, we proposed a custom made iterative floating point FPGA based architecture for matrix inversion using Cholesky factorization. The underline plan of implementation is accelerating matrix inversion of positive definite (Hessian matrix) system in QP solvers to achieve high throughput and to improve area/speed metric to meet real time demands of QP solvers used in Model Predictive Controllers. Our design offers a trade-offs between area and speed for different data

width problems while performing matrix inversion using FPGA's. Use of LNS approach for single precision ANSI/IEEE-754 floating point square root and division arithmetic not only improves hardware utilization but also gives better throughput compared to conventional FP implementation.

The remainder of the paper is organized as follows: section two describes Cholesky algorithm and its associated computational complexities. Proposed FPGA based four PE iterative architecture is explained in section three. For floating point implementation, LNS based algorithm is described in section four. Section five analyses the performance of our implementation with previously proposed architectures. Section six concludes the paper.

2. CHOLESKYFACTORIZATION

Cholesky factorization (LL^T) is one of the alternatives to Gauss-Jordan elimination method for solving dense linear systems i.e.$Ax = b$, where A is a $N\,x\,N$ positive definite matrix and b is called as right hand side. It is also used to find inverse of positive definite matrix. If $A \in R^{nxn}$ is symmetric and positive definite, then it can be factored as

$$A = L\,L^T \quad (1)$$

Where, L is lower triangular and nonsingular with positive diagonal elements. Inverse of matrix *A* is obtained by

$$A^{-1} = (L^{-1})^T L^{-1} \quad (2)$$

The diagonal elements of lower triangular matrix ***L*** are computed as

$$Lii = \sqrt{A_{ii} - \sum_{k=1}^{i-1} (L_{ik})^2} \qquad (3)$$

The non- diagonal elements of *L* matrix are given by

$$Lij = \frac{1}{L_{i,i}} [A_{ij} - \sum_{k=1}^{i-1} (L_{ik} * L_{jk})] \qquad (4)$$

$$for\ j = i+1, i+2 \ldots\ldots\ldots\ldots N$$

Cholesky factorization is the computationally most expensive step in numerically solving a positive definite system. From Eq. (3) and (4) it is clear that, due to inherently recursive computation process and associated floating point division and square root operations in Cholesky factorization, it is very difficult to obtain hardware acceleration, but factorization allows computing diagonal elements(L_{ii}), and non-diagonal elements (L_{ij}) of *L* matrix in parallel. The equations shows input matrix *A* gets recursively modified during *N*computational step in order to obtain lower triangular matrix, *L*.

3. FPGA IMPLEMENTATION

This section details the custom architecture developed for matrix inversion using Cholesky factorization with IEEE-754 single precision floating point data format. The key important feature of proposed architecture includes:

- Iterative architecture using four processing elements (PE's) throughout the design
- Use of LNS approach for IEEE-754 single precision floating point square root and division algorithm
- By exploiting computational parallelism, for NxN matrix inversion, architecture uses floor $[(N+2)^2/8]$clock cycles compare to conventional ***O(1/3N^3)*** clock complexities

The proposed four Processing Elements basediterative matrix inversion architecture uses unit element i.e. PE,which consists of multiplier block, adder block, one register and sequential control logic block, which controls the data feed operation to multiplier,adder and register as shown in figure 1.

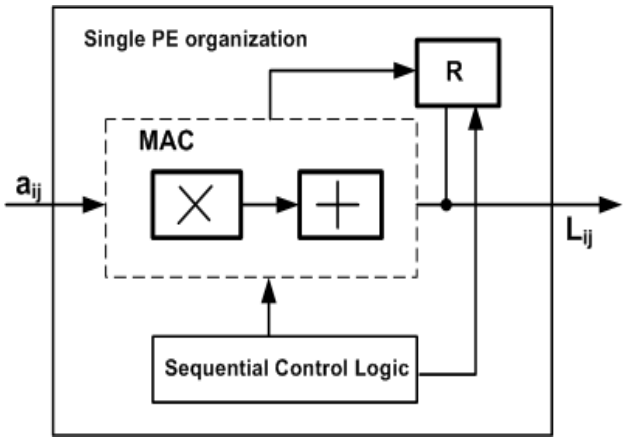

Fig. 1 Structure of Single PE

Assume that for matrix *A* of size (NxN), whose inverse has to obtain is stored in internal RAM. To calculate diagonal and non-diagonal elements of resultant matrix *L*, The flow of data arranges in such a way that, input to the PE's are the diagonal and non- diagonal elements of input matrix *A* with non-diagonal and triangular elements of resultant matrix *L* operated in iteratively Fig. 2 shows proposed iterative architecture of four PE's that have been used in realization of matrix inversion algorithm.

Form eq. (3) and (4), the diagonal elements(L_{ii}), indicates the square root of summation of square of non- diagonal elements of the same row of *L* matrix. The non-diagonal elements (L_{ij}) are obtained by summation of product of two non- diagonal elements obtained from previous iterations and divided by diagonal elements of *L* matrix. For FPGA implementation, we proposed an iterative architecture using Diagonal Processing Unit (DPU), Non-diagonal Processing Unit (NDPU) and Triangular Processing Unit (TPU) as a computationalprocessing units to calculate diagonal, non-

diagonal and triangular elements of resultant matrix *L* respectively.

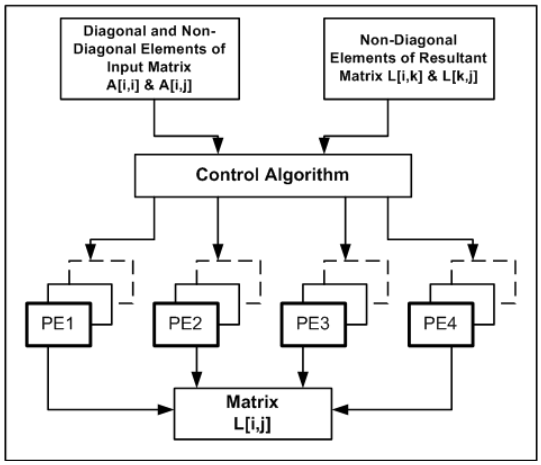

Fig. 2 4-PE Structure of Propsed Iterative Matrix Inversion Architetcure

3.1 DPU

The non-diagonal elements of decomposition matrix ***L***and diagonal elements of input matrix ***A*** are imported to DPU in parallel. The DPE, first calculates summation of square of non-diagonal elements of that row subsequently subtracting the resultant square sum from diagonal elements of input matrix ***A***computing its square root using proposed LNS algorithm, produces the resultant diagonal elements of ***L***matrix.

Fig. 3 shows hardware architecture of DPU for Cholesky factorization

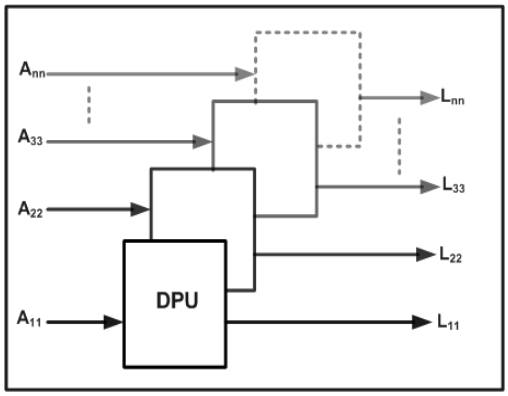

Fig. 3 Hardware architecture of DPU for Cholesky Factorization

For a matrix of size of 4*x4* it utilizes six multipliers, three adders and three subtractors with a square root operator. In general, DPU can process for *NxN* matrix iteratively with *[(N)*(N-1)/2]* multipliers, *[(N-1)*(N-2)/2]* adders, *(N-1)* subtractors and a square root operator

3.2 NDPU and TPU

An iterative architecture for NDPU and TPU are shown in fig. 4. The adjacent non-diagonal elements of matrix *L* are fed parallel to TPU which computes product of these non-diagonal elements of same column of *L* matrix and processes to NDPU

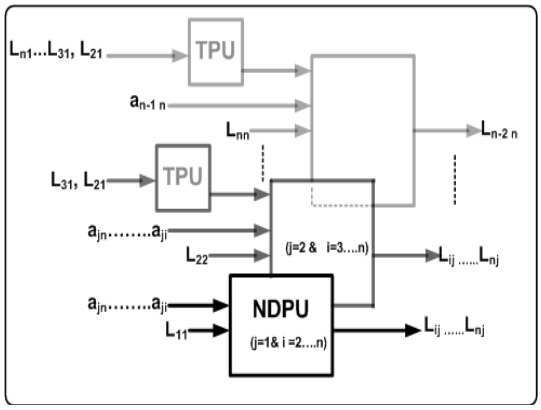

Fig. 4 Iterative architecture of TPU and NDPU

For matrix of size *4x4*, it takes four multipliers and three adders. To be specific, for *NxN* matrix it will execute using *(N)* multipliers and *[(N-2)*(N-1)/2]* adders. The non-diagonal elements of input matrix *A* and results of TPU both are fed parallel to NDPU. Using *(N)* subtractors and *(N)* multipliers NDPU calculates non-diagonal elements of factored *L* matrix of size *NxN*. The generalized iterative architecture for TPU and NDPU is shown in fig. 5.

The overall clock latency of Cholesky factorization to obtain inverse of $N \times N$ positive definite input matrix is given by $\text{floor}[(N+2)^2/8]$; where, N is size of input matrix. The problems of sparsity in case of sparse matrix are solved by preconditioning the input matrix.

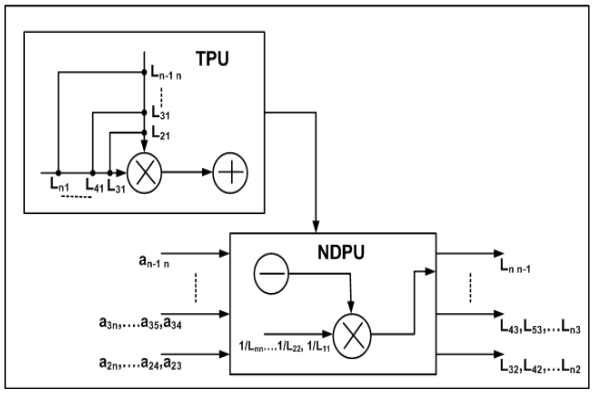

Fig. 5 Hardware Architecture of NDPU and TPU

Fig.6 shows data dependency graph (DG) for Cholesky factorization to obtain lower triangular matrix *L* by considering N=4. The graph shows that, the elements of lower triangular matrix *L* are obtained in four clock cycles using four PE'. As matrix size increases the size of triangular arrays also increases leads to more clock cycles to obtain elements of lower triangular matrix *L*.

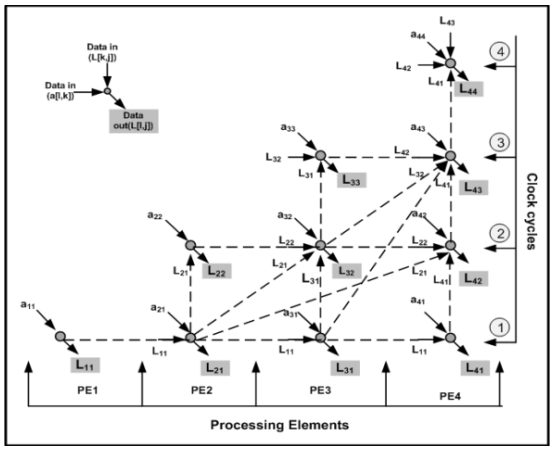

Fig. 5 Data dependency (DG) graph (N=4)

3.3 Tradeoff for selection of PE's

In case of square rational number matrix, we get N/4 times Nx4 matrix blocks. This holds good for matrix of size 4x4 and more. In case of odd number of matrix size results in N/4 times Nx4 matrix blocks and to get remaining elements of output matrix we have to reorder the PE's subject to remaining columns of matrix A. The PEs' get effectively utilized when number is limited to four. If we select any number of PE's which are greater than four, then the utilization of PE's are reduces drastically as function of increase in matrix size. 4 PE's are optimistic for matrix size 4x4 and more that holds tradeoffs between execution time and resource utilization. If we select PE's more than 4, underutilization increases exponentially as matrix size increases. But selection of PE's less than 4 (i.e. 3 or 2), underutilization reduces compared to 4-PE architecture but as matrix size increases more than 4 due to iterative nature of architecture clock cycles increases. The figure clock cycle*resources comes to be optimize for 4-PE architecture than any other number of PE's.

4. LNS APPROACH

Form Eq. (3) and (4),main computational task involved in Cholesky factorization is to obtain floating point square root and division. For implementation of these arithmetic, instead of conventional methods like piecewise polynomial approximation, Newton's iterative algorithm, or Goldschmidt's algorithm, we propose a parametric computational logarithmic (LNS) based algorithm which uses a simple logical shift and compare logic with look up table (LUT)

Logarithmic numbers can be viewed as a specific case of floating point numbers where the mantissa is always 1, and the exponent has the fractional part explained by [11]. The approach suggested in our work, uses single precision floating point number with 32-bit data width represented as follows:

Sign Bit	**Fixed Point Logarithmic Value**	
S	Integer: M bits	Floating point: F bits
1	7	24

In LNS system, number is expressed as $(-1)^{S} \times 2^{M.F}$ which has a similar representation as floating point number. Using LNS approach, the basic arithmetic operations are translated into hardware resource optimized architecture using simple addition, subtraction, left shift and right shift operators. Let *a* and *b* be the logarithmic representations of numbers A and B, respectively (provided A> B), then basic LNS operations of these two numbers [11] are shown in table 1

Table1.Arithmetic Operations Using LNS

Arithmetic Operation	**Logarithmic Representation**
A × B	log2 (a) + log2 (b)
A / B	log2 (a) -log2 (b)
A 2	2 log2 (a))
$\sqrt{A}$	log2 (a) / 2

Conversion of floating point number to equivalent LNS is performed by dividing the range of given numbers for three different cases as: Number less than 1, number greater than 2 and number in between 1 and 2. If the number is less than 1, logarithmic equivalent of such numbers are negative. Integer part of negative number is obtained by recursive left shift until number lies between 1 and 2 and fractional part corresponding to logarithmic equivalent is fetched from LUT. Finally, logarithmic equivalent of given FLP number is represented in two's complement format by setting sign flag bit 1 and combining number of shifts (count value) with corresponding log values form LUT.

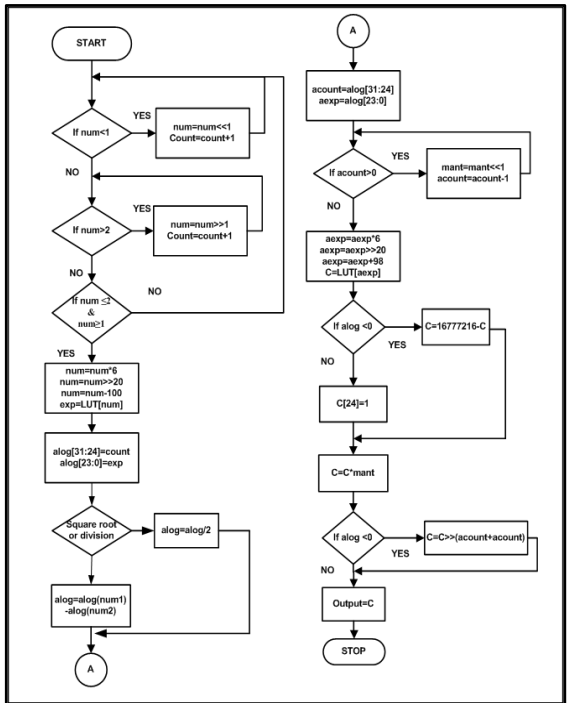

Fig. 6 32-bit square root and division algorithm using LNS approach

If number is greater than 2, integer part is obtained by recursive right shift until number lies between 1 and 2, for representation of number in its logarithmic equivalent the procedure explained above is repeated. The details of the algorithm are explained using flow chart as shown in fig. 6. As the data width increases the time required for LNS conversion also increases due to more number of shift and add operations. After testing for different data bit widths, the approach is proved to be better than conventional approaches up to IEEE single precision Floating Point Number (32-bit) for its accuracy and clock latency.

5. EXPRIMENTAL RESULTS AND ANALYSIS

The proposed architecture is implemented using Xilinx-Virtex4 FPGA ((XC4VSX35) operating at 125MHz. The algorithm is described in Verilog using Xilinx ISE 11.4. Testing and validation is performed using MATLAB's System Generator for different size of matrices using single precision floating point data format. Table 2 shows design summary of resource utilization of proposed architecture for 40 x 40 single precision floating point matrix inversion.

Table2.Design Summary of Resource utilization

Resources	Utilization
Device	Virtex-4(XC4VSX35)
Clock	125MHz
Data width	32-bit floating point
Slices	2993(19%)
Flip/flops LUT's DSP-48	791(2%) 4993 (16%) 136 (70%)
BRAM	2(1%)

Table 3 shows summary of resource and time utilization of proposed architecture for different matrix sizes. An experimental result shows that the resources utilization of proposed architecture is independent of matrix size at particular clock frequency. The computation time increases as a function matrix size and latency used to map the design.

Table 3.Summary of Resource and Time Utilization of Proposed Matrix Inversion Architecture

Matrix size	CLB's	Clock MHz	Computational Time (µs)	Throughput (Mupdates/s)
4x4	748	125	0.032	31.25
6x6	748	125	0.064	15.62
20x20	748	125	0.480	2.083
40x40	748	125	1.760	0.568

The validity and scalability of proposed architecture is tested for different size of matrices, the graph shows relation between clock cycles utilization for different size of matrices.

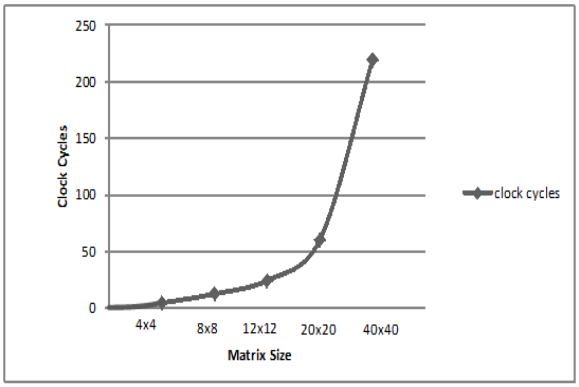

Fig. 7 Clock cycles vs. matrix size

The performance of proposed architecture is compared with previously reported designs by normalizing their results for 128 x128 single precision floating point matrix inversion. Table 4 shows the performance comparison of different Cholesky factorization based matrix inversion architectures. It is seen that, proposed architecture archives improvement in computational time compared to previously reported designs.

Table 4.Performance Comparison of 128 x128 Matrix Inversions for 32-bit Floating Point Data Format

Design	Device	Achievable Clock (MHz)	Computational time (µS)
Depeng Yang [5]	XC4VLX200	245	35.529
Depeng Yang [3]	XC6VSX475	180	173.25
Proposed	**XC4VSX35**	**125**	**16.90**

6. CONCLUDING REMARKS

This paper has investigated the hardware performance of custom architecture developed for Cholesky factorization as a computing platform used to find inverse of positive definite symmetric matrix. Throughout the design process, the underlined plan was to design an architecture that is more effective in both area and clock cycle utilization. The use of LNS approach for floating point square root and division arithmetic over conventional FP number system and an iterative custom design architecture using DPU, TPU and NDPU as a processing units the proposed architecture achieves better throughput compared to previously reported designs. The final design, after synthesis using Xilinx ISE 11.4, has been downloaded into Vertex-4 FPGA operating at 125MHz. Its functionality and scalability has been verified through HIL co-simulation for different size of matrices.

7. REFERENCES

[1] Antonio Roldao and George A. Constantinides, "A High Throughput FPGA-Based Floating Point Conjugate Gradient Implementation for Dense Matrices*", ACM Transaction on Reconfigurable Technoogy Systems*, Vol. 3 (1), Jan. 2010.

[2] J. Sun, et.al, "High Performance Mixed-Precision Linear Solver for FPGAs," *IEEE Transaction on Computers*, vol. 57, (12) Dec. 2008

[3] D. Yang**,** et. al, "Performance Comparison of Cholesky Decomposition on GPUs and FPGAs," *Symposium on Application Accelerators in High Performance Computing (SAAHPC), knoxville*, TN, 2010.

[4] D. Yang, G. D. Peterson, and H. Li, "High Performance Reconfigurable Computing for Cholesky Decomposition," *Symposium on Application Accelerators in High Performance Computing (SAAHPC*), UIUC, July, 2009.

[5] D. Yang**,** et.al, "An FPGA Implementation for Solving Least Square Problem," *The 17th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Napa, California, April. 2009.

[6] Antonio Roldao Lopes and George A. Constantinides, "A High Throughput FPGA-Based Floating Point Conjugate GradientImplementation", *Proceedings of the 4th international workshop onReconfigurable Computing: Architectures, Tools and Applications*, Springer-Verlag, Berlin, Heidelberg,2008, pp 75-86

[7] O. Maslennikow, et.al, "Parallel Implementation of Cholesky LL^T-Algorithm in FPGA-Based Processor," *Proceedings of the 7th international conference on Parallel processing and applied mathematics Springer-Verlag Berlin Heidelberg*, , June-2007pp. 137-147

[8] Maslennikow,et.al.,"Implementation of Cholesky LL^T-Decomposition Algorithm in FPGA-Based Rational Fraction Parallel Processor," Mixed Design of Integrated Circuits and Systems, *14th International conference on mixed design (MIXDES 2007)*,Ciechocinek, POLAND, 21-23 June 2007, pp.287-292

[9] Burian, A. et.al , "A fixed-point implementation of matrix inversion using Cholesky decomposition," Micro-NanoMechatronics and Human Science, *Proceedings of the 46th IEEE International Midwest Symposium on* In Circuits and Systems, 2003. MWSCAS , Dec. 2003, pp. 1431- 1434

[10] S. Haridas, "FPGA Implementation of a Cholesky Algorithm for a Shared-Memory Multiprocessor Architecture,", A masters thesis, submitted to the faculty of New Jersey Institute of Technology, May 2003.

[11] Garcia et al. "LNS Architectures for Embedded Model PredictiveControl Processors"*In Proceedings of International Conference onCompilers, Architecture, and Synthesis for Embedded Systems,CASES 2004, Washington DC, USA, Sep. 2004*

[12] A book on "VLSI Digital Signal Processing System Design and Implementation", by K. K. Parhi, Wiley, 1999.