Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
2326-9865

FPGA Implementation of UaL Decomposition, an alternative to the LU factorization

Sai Ruchitha, Ramesh Chinthala
Department of Electronics and Communication Engineering,
Amrita School of Engineering, Bengaluru

Amrita Vishwa Vidyapeetham, India
ruchithasvs2014@gmail.com, c_ramesh@blr.amrita.edu

Article Info
Page Number: 1081-1094
Publication Issue:
Vol. 71 No. 4 (2022)

Article History
Article Received: 25 March 2022
Revised: 30 April 2022
Accepted: 15 June 2022
Publication: 19 August 2022

Abstract—Matrix decomposition is an important method used in many applications such as circuit simulations, for example Modified Nodal Analysis (MNA matrices), and in communication systems, for example to find minimum mean square error (MMSE) in MIMO systems for detecting the transmitted symbol vector from the received symbol vector. In this paper, an FPGA based hardware implementation of an alternative solution to LU factorization technique called UaL decomposition method is proposed. The RTL code of the UaL algorithm is developed and simulated using Xilinx Vivado software. The RTL code of the proposed FPGA based UaL decomposition hardware architecture is synthesized by targeting Virtex-5 FPGA which supports the data input in single-precision Floating-point representation format. The FPGA implementation of the UaL decomposition method is compared with the existing FPGA implementations of LU, LDL, Cholesky and QR decomposition methods in terms of area, frequency and computational time**.** The proposed sequential FPGA implementation of UaL decomposition utilizes 47% less resources than the existing best parallel LU factorization FPGA implementation but requires 50% more computational time, and operates at 210 MHz which is approximately three times than the operating frequency of best existing decomposition implementation (LU decomposition). The parallel implementation of UaL decomposition is expected to reduce the computational time by 32% compared to sequential UaL and 68.9% compared to LU decomposition.

Keywords— LU factorization, UaL decomposition, FPGA, QR, LDL, Cholesky, LKU.

I. INTRODUCTION

Matrix calculations are a primary segment of most logical processing problems. As computers are having a restricted precision, solving complicated operations related to matrices is not that efficient. One such calculation is matrix decomposition method. This decomposition method breakdown a single complex matrix into two simpler matrices which results is performing more complex operations in a simple way. The other name for this matrix decomposition is matrix factorization, which is a basis for linear algebra in personal computers and also for some common operations like system of linear equations solving's, finding inverse and determinant of a matrix.

One specific such matrix decomposition method is LU factorization technique, which disintegrates a matrix into a result of two simpler matrices called a lower triangular matrix and an upper triangular matrix. This LU computation is a significant advance in tackling enormous systems of linear equations [1]. Numerous implementations are having this LU factorization as a centre portion which should be used frequently. From the literature it can be found that the LU decomposition is good in terms of computational speed compared to other existing matrix decompositions methods such as QR, LDL and Cholesky [20].

The precision of the resultant L and U matrices can be improved by reducing the round off errors. UaL decomposition algorithm is the modified version of LU decomposition algorithm which achieves computational accuracy compared to the existing LU decomposition algorithms by reducing the round-off errors [3].

In this paper, the FPGA implementation of UaL factorization algorithm is put-forwarded and implemented on Virtex-5 FPGA. As the regular processors, for example, CPU confronted an incredible difficulty due to its high delay correspondence in its cores, low effectiveness in parallel computations and less memory data transmission. Subsequently, FPGA turns into a hopeful stage to accelerate the factorization of a matrix because of its bountiful logic resources and incredible parallel computing capacity [2]. The proposed UaL implementation on FPGA shows the reduction in LUT resources compared to FPGA implementation of LU decomposition as it requires 1) no extra circuitry to eliminate round-off errors which are caused by divisions in LU 2) due to the simple calculations as UaL does not use complex-valued multiplications. Computation time more as UaL is sequential, but the parallel implementation of UaL improves the computational time, when same parallelism as UL is used UaL architecture.

This paper is structured in following way: Section II explains about background and related work of LU based FPGA. Section III gives the details of the UaL decomposition algorithm. Section IV is about the implementation of FPGA based UaL decomposition algorithm. Section V shows the results and comparison with other related algorithms. Conclusions are in Section VI.

II. BACKGROUND AND RELATED WOTK

As discussed in previous section, LU decomposition of a matrix will result in two matrices is shown in below Figure. 1, where we can see an input matrix A is factorized and results two output matrices called L and U in product form and this can be symbolized as A=L×U, Where L represents a lower triangular matrix and U represents an upper triangular matrix.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Fig. 1. LU decomposition of a 3x3 matrix

Here, a_{11}, l_{11}, u_{11} denotes the matrix elements for each A, L and U matrices [4].

A plenty of analysis is made on LU decomposition using FPGA's. In [5] a new Architecture which is efficient in area and Throughput is proposed by making changes in an existing algorithm called KLU for LU decompositions. An algorithm for an architecture with multiple processors for a parallel block LU decomposition with a column of two stage was developed in [6] by partitioning the given matrix into different larger blocks which are then again breaks down into smaller blocks equal to the count of processors. In [7] the computation accuracy of the fixed-point based architecture of LU decomposition and the factors affecting the accuracy are studied. In [8] a parallel recursive algorithm for LU factorization was developed based on the Divide and Conquer paradigm.

In [9] stochastic multiplier and divider designs are proposed for designing a Lower-Upper decomposition (LUD) scheme. In [10] an efficient cache architecture for an FPGA based parallel sparse LU factorization method based on Gilbert-Peierls (GP) algorithm was proposed. In [11] for decomposing dense matrices using LU decomposition an approach called OpenMP task was developed in parallel method which is based on each tasks that are occurred during LU decomposition in block wise.

In [12] for the computations and simulations of circuits a GPU based solver using sparse LU was proposed. This proposed method is known as GLU method which means, GPU accelerated LU decomposition method, that depends on LU decomposition algorithm of hybrid type right looking. In [4] an architecture to improve LU decomposition computations was proposed in which they have used a same PE (processing element) multiple times by utilizing pipelining techniques, that makes this architectural design to be more efficient in terms of resource utilization and also makes this available for resources that have limitation in hardware and for some requirements in real time.

In [13] an architecture to accelerate sparse LU factorization was designed by modifying the architecture that is in dataflow model to hold up with heterogeneous PE's and also design networks with dual channels which reflects its properties in LU decomposition graphs. In [14] Breaking Sequential Dependencies in FPGA-based Sparse LU Factorization was shown by using depth-limited substitution, and reassociation of the resulting computation. In [15] FPGA implementation using a dataflow model is done for LU factorization method along with a task separation and algorithm assignment based on a modified Kernighan-Lin is also presented.

Two architectures are proposed in [16], one is for FPGA based LU factorization for limited sized matrices with single precisions and the other is for block level multiplication of block LU factorization having larger matrix sizes. In [17] author shows an algorithm based on left looking is far better than an algorithm of right looking type and then implemented an architecture in parallel way using the algorithm of left looking type for sparse LU factorization algorithm on FPGA. And a dependence study was used in performance of column operations.

In [18] an LU factorization architecture using FPGA having higher performance along with efficient memory was implemented by applying a few transformations in series, that includes blocking loops and mapping of space and time, on non-blocking sequential LU factorization

and also consists of PE's in linear fashion, for implementing block LU factorization algorithm. In [1] FPGA architecture having higher performance for block LU factorization was implemented by combining the arithmetic units of floating points along with the access patterns of memory which makes easier to cover the access latency of memory behind computations.

Using Gaussian Elimination theory analysis, a parallel FPGA based LU factorization design with higher performance is implemented [19].

III. UaL Decomposition

In [3] Hashemian proposed a novel algorithm called UaL decomposition algorithm for nodal analysis of a circuit, whose matrix representation of the circuit [21, 22] is shown below in equation (1)

$$YV = J \qquad (1)$$

In above equation, Y represents an admittance matrix in nodal analysis, the voltages of nodes are represented in V matrix and the stimuli of the total nodal in vector form is represented in J (both voltage and current sources). In this decomposition the admittance matrix Y is partitioned into two higher and lower triangular matrices U and L and can be given by (2).

$$Y = LU \qquad (2)$$

$$LUV = J \qquad (3)$$

Now, from above equation L has to be moved to other side of the equation to make it balance and this is shown in below equation (4).

$$UV = LJ \qquad (4)$$

When the equations (2) and (4) are compared, these will have the same U matrix but the matrix L in both the equations are in inverse with one another. For the sources of current and for the voltages of nodes the arrays are represented as $J = [j1, j2, \ldots, jn]^t$ and $V = [v1, v2, \ldots, vn]^t$ respectively and when the (4) is solved using these two arrays it results in following equation (5).

$$v_i = \frac{1}{u_{ii}} \left(\sum_{k=1}^{i} l_{ik} j_k - \sum_{l=i+1}^{n} u_{il} v_l \right) \qquad (5)$$

Where i ranges as n, n-1, … , 1 and the variables u_{ii}, u_{il}, and l_{ik} represents the elements of the matrices L and U. To get the voltages of the nodes the necessary divisions required are done by using u_{ii} in (5). Hence it can be seen that the operations in UaL factorization does not require any divisions, which helps in eliminating pivots of the matrix. Features which differentiates (4) from equation (2) are pointed, explained and are proved in [3]. The main equations involved in determining the elements of the matrices L and U are mentioned in the equations (6), (7).

$$u_{jk}^{(y)} = u_{ii}^{(i)} u_{jk}^{(i)} - u_{ji}^{(i)} u_{ik}^{(i)} \qquad (6)$$

$$l_{jk}^{(y)} = u_{ii}^{(i)} l_{jk}^{(i)} - u_{ji}^{(i)} l_{ik}^{(i)} \quad (7)$$

Where k takes all values, the superscript variable i specifies the levels of columns processing and y is assigned to i+1.

However, it must be noted that the UaL computational accuracy is more compare to the LU decomposition as division operations are removed or made with zero remainders and the round off errors are eliminated totally. The pseudo code of UaL factorization algorithm is given in Algorithm I [3]:

Algorithm I UaL factorization

```
Initialize U = Y, L = I
for i = 1: n-1
  for j = i + 1: n
  /* Already zero (sparsity) */
    if U (j, i) = 0
      for k = i + 1: n
        U (j, k) = U (i, i) U (j, k)
        if (i > 1)
        /* pivotal elimination */
          U (j, k) = U (j, k)/U (i-1, i-1)
        end if
      end for k
      for k = 1: j
        L (j, k) = U (i, i) L (j, k)
        if (i > 1)
        /* pivotal elimination */
          L (j, k) = L (j, k)/U (i-1, i-1)
        end if
      end for k
    end if
    /* nonzero terms */
    else
      for k = i + 1: n
        U (j, k) = U (i, i) U (j, k) – U (j, i) U (i, k)
        if (i > 1)
          U (j, k) = U (j, k)/ U (i-1, i-1)
        end if
      end for k
      for k = 1: j
        L (j, k) = U (i, i) L (j, k) – U (j, i) L (i, k)
        if (i > 1)
          L (j, k) = L (j, k)/ U (i-1, i-1)
        end if
      end for k
    end else
  end for j
end for i
```

By utilizing the UaL decomposition algorithm as shown above in C++ language for **MNA** matrices which is proposed in [3], we have proposed hardware architecture and implemented UaL Decomposition on FPGA as UaL decomposition is expected to be better than the LU decomposition in terms of accuracy.

IV. FPGA BASED UAL DECOMPOSITION

This section provides the details of the proposed hardware architecture design of UaL decomposition and its functionality in performing UaL Decomposition on FPGA.

Fig. 2 shows the top-level architecture mainly that consists of three units namely; memory subsystem unit, control unit and compute unit.

The memory subsystem consists a U memory and L-memory. The U memory is initially used to store the input matrix 'A' data and then matrix A data is overwritten with computed values of U matrix as the computation progresses. The L memory initially stores Identity matrix values and later gets updated with the L matrix values as it gets computed.

The UaL compute unit consists of four multipliers M1, M2, M3, M4, two subtractors, S1, S2, and two dividers D1, and D2. These arithmetic units and the memory units are connected through Address and Data buses as shown in Fig. 2. The detailed internal connections are shown in Fig. 3 which shows the data flow during the computation as well.

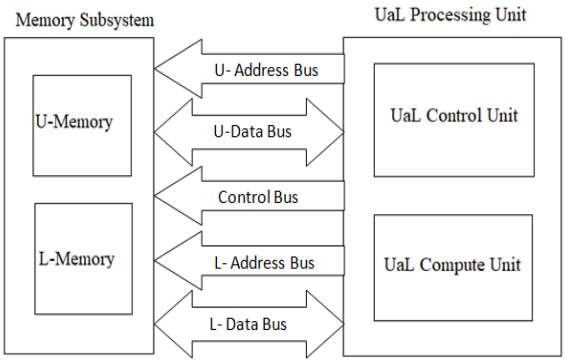

Fig. 2: Top level architecture of UaL

Control unit controls the matrix data flow between the memory sub system unit and the UaL compute unit also controls the sequence of single precision floating point arithmetic calculations as per the UaL algorithm and then the output values are stored back into the memory subsystem.

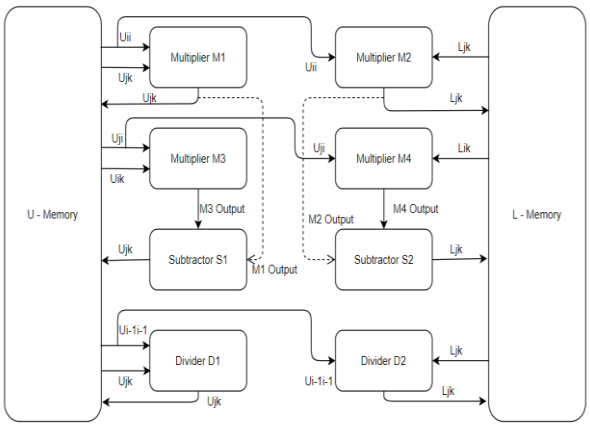

Fig. 3: Data flow during the calculations of U and L matrix values.

Fig. 3 shows the data during the calculations of U and L matrix values as per the UaL algorithm. According to the algorithm, a set of operations are done on Input matrix and Identity matrix to obtain the elements for final U and L matrices. Two sets of operations are done based on whether Uji element is zero or non-zero.

If Uji is zero, first and second for loops in the algorithm are used to update U and L matrix elements respectively. Considering the first for loop, the U matrix elements are computed by passing Uii and Ujk as inputs to the Multiplier M1 and output is used to updated the Ujk element. Based on the value of i, if i is greater than 1 then this updated Ujk is passed to the divider D1 as one of the input along with the other input element Ui-1,i-1, then the divider D1 output is used to update the Ujk element. Considering the second for loop, the L matrix elements are computed by passing Uii and Ljk as inputs to the Multiplier M2 and output is used to update the Ljk element. Based on the value of i, if i is greater than 1 then this updated Ljk is passed to the divider D2 as one of the input along with the other input element Ui-1i-1, the output of the divider D2 is used to update the Ljk element. Therefore, if Uji is zero only two multipliers and two dividers are required to compute/update the U and L matrix elements.

If Uji is not equal to zero, third and fourth for loops in the algorithm updates the U and L matrix elements respectively. Considering the third for loop, the U matrix elements are computed by passing Uii and Ujk as inputs to the Multiplier M1 and Uji and Uik as inputs to Multiplier M3 and these M1, M3 outputs are passed as inputs to Subtractor S1 whose output is used to update the Ujk element. Based on the value of i, if i is greater than 1 then this updated Ujk is passed to Divider D1 as one of the input along with the other input element Ui-1i-1, then the output of divider D1 is used to update the Ujk element. As per the fourth for loop in the algorithm, the L matrix elements are computed by passing Uii and Ljk as inputs to the Multiplier M2 and Uji and Lik as inputs to Multiplier M4 and these M2, M4 outputs are passed as inputs to Subtractor S2 whose output is used to update the Ljk element. Based on the value of i, if i is greater than 1 then the updated Ljk is passed to Divider D2 as one of the input along with the other input element Ui-1i-1, the output of the divider D2 is used to update the Ljk element. Therefore, if Uji is not zero then all the arithmetic units in the compute unit are in active state.

The input and output data are in the format of IEEE - 754 floating point Single Precision representation, therefore, 32-bit floating point multiplier, subtractor and divider are used to perform the arithmetic calculations on the input matrix data.

V. Results and Discussions

The RTL code of UaL Algorithm is described using Verilog to model its hardware architecture and is implemented targeting FPGA Virtex 5 board to find the resource utilization, and operating frequency, and power consumption using the Xilinx Vivado.

Initially the input matrix A, and identity matrix I are stored in U and L memories respectively. The content of these memories is verified by displaying using simulations as shown in Fig. 4 and 5. UaL decomposition is performed on those two matrices as per the UaL decomposition algorithm. During the algorithmic flow the computed values of U and L matrices are stored in U and L memories respectively. All the arithmetic calculations are performed by using subtractor, divider and multiplier floating point sub-modules. Each sub module is simulated and synthesized separately by applying floating point data. The functionality verification of these floating-point arithmetic units is done with the help of simulation waveforms as shown in Fig. 6, 7, and 8. Later all these sub modules are instantiated according to the UaL decomposition algorithm-based architecture as shown in Figure 3 and then a testbench is written that uses floating point data to drive the UaL decomposer and simulated to verify its functionality. Figure 9 shows simulation results of the UaL decomposition of 4x4 matrix.

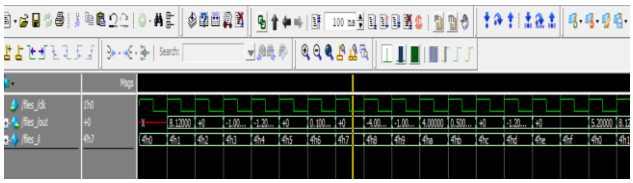

Fig. 4: Storing input matrix initially in U memory

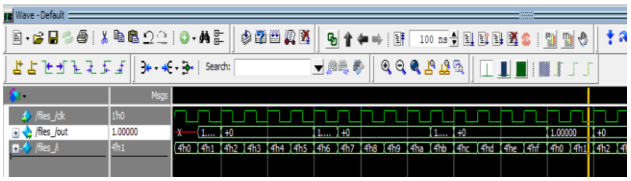

Fig. 5: Storing Identity matrix initially in L memory

Figures 4, 5 shows that U and L memories that are filled initially with input and identity matrix values respectively.

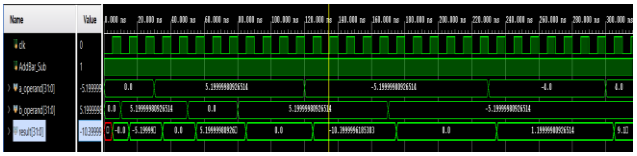

Fig. 6: Floating Point Subtraction of two numbers

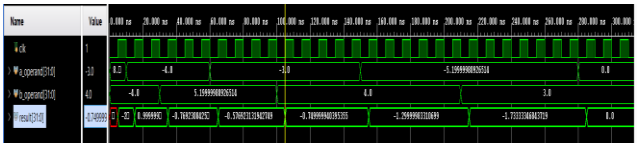

Fig. 7: Floating point Division of two numbers

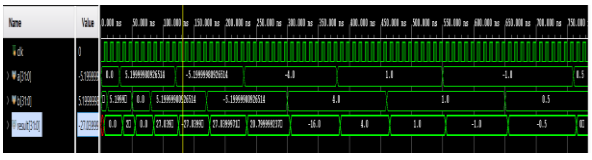

Fig. 8: Floating Point Multiplication of two numbers.

Figures 6, 7, 8 shows the floating-point subtraction, division and multiplication of two numbers for 32 – bit single precision floating point data.

Fig. 9 represents the simulation output of UaL decomposi-tion for 4x4 matrix. The input matrix taken for simulation is from [3].

The Table 1 shows the count of different types of arithmetic operations required by each type of decomposition algorithms for decomposing a given matrix into L and U matrices. From Table 1 it can be noticed that compared to

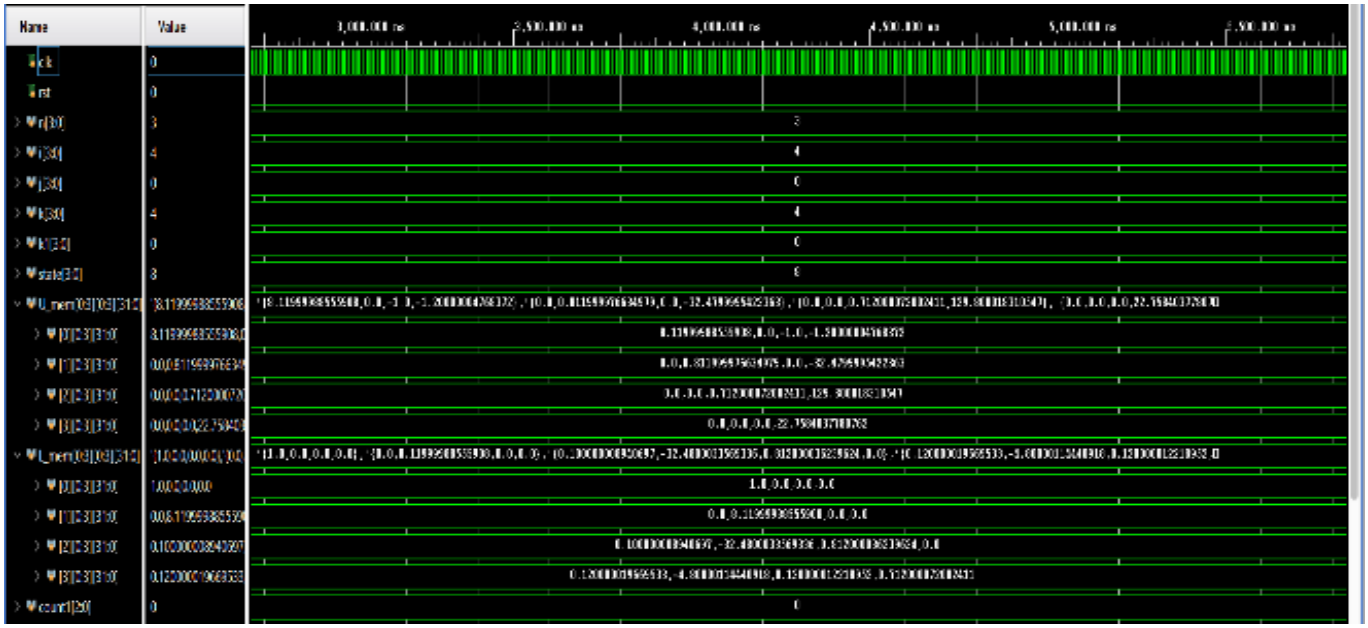

Fig. 9: UaL Decomposition of 4x4 matrix

TABLE I: NUMBER OF OPERATIONS REQUIRED FOR DIFFERENT DECOMPOSITION ALGORITHMS FOR A 4X4 MATRIX.

Arithmetic Operations	QR	LU	LDL	Cholesky	UaL
Addition	89	12	0	0	0
Subtraction	0	0	5	5	23
Multiplication	16	21	16	11	57
Complex- valued Multiplication	49	9	3	3	0
Division	16	3	2	2	16
Square root	4	0	0	2	0
Total no. of Operations	174	45	26	23	96

TABLE II: CHARACTERISTICS OF 32-BIT FLOATING POINT UNITS USED IN UAL FPGA IMPLEMENTATION

Resource Utilizations	Subtractor	Multiplier	Divider
Pipelining stages	6	5	3
Slice Registers	332	138	305
Slice LUT	680	109	391

Fully used LUT- FF pairs	284	87	226
BUFG	1	1	1
DSP48	-	2	4
Frequency (MHz)	246.591	211.551	211.551

other decomposition methods, UaL decomposition does not require any complex operations like complex-valued multiplications and square roots to complete its decompositions calculations. Also, as pipelining techniques are not used in UaL decomposition architecture, it has required more number of operations to complete its decomposition process whereas the comparison methods are pipelined [20].

Table II shows the resource utilizations of different floating-point units used in the proposed UaL processor. It also shown that subtractor unit is having more pipelined stages hence having high frequency when compared with

TABLE III: FPGA RESOURCE UTILIZATION FOR 4X4 MATRIX

FPGA Resource	**QR**	**LU**	**LDL**	**Cholesky**	**UaL**
Total Slices	8287	4606	2469	2027	-
Slice Registers	18256	10711	5325	5247	2554
Slice LUTs	21268	10022	5006	4891	5235
Block RAMs	4	4	2	2	-
DSP48s	224	57	28	23	8

multiplier and divider. The overall operating frequency is dominated by the module with low operating frequency, hence the proposed UaL operating at 210 M Hz.

Table III shows the comparison of the resource utilizations of different decomposition algorithms based FPGA implementations. As the existing methods like QR, LU, LDL, Cholesky [20] hardware implementations use parallel execution of matrix operations, and hence required more number of resources. Also shown that the proposed UaL design requires less number of LUT's as matrix elements are processed one element at a time, and pipelined arithmetic units are used hence the operating frequency is more. Also operations like complex-valued multiplications and square roots are avoided, resource utilization of UaL algorithm is decreased.

The Table IV shows the comparison of computation time of UaL factorization method of 4x4 matrix with the existing factorization techniques like LU, QR, LDL, Cholesky that are referred from [20]. Calculation of UaL computation time was done by dividing the number of clock cycles required with the maximum operating frequency. From the Tables III and IV we can observe that, as LU based architecture has low computational time but utilizes more resources and the UaL based architecture requires 47% less number of resources and 50% more computation time compared to LU. Therefore, it can be concluded that there is a trade-

off between resource utilization and computation time and the UaL algorithm can be used in applications which require more accuracy.

TABLE IV: COMPUTATION TIME OF 4X4 MATRIX ON DIFFERENT MATRIX DECOMPOSITION HARDWARE ARCHITECTURES.

	QRQ	QRR	LUL	LUU	LDL	CHOL	UaL
No. of clock cycles	59	75	22	20	44	63	258
Max freq (MHz)	76.6	65.5	78.6	67.7	73.1	71.8	**210.06**
Computation time (us)	0.77	1.15	0.28	0.30	0.60	0.88	1.23

TABLE V: COMPUTATION TIME OF 4X4 MATRIX ON DIFFERENT MATRIX DECOMPOSITION HARDWARE ARCHITECTURES (PARALLEL ESTIMATION).

	QRQ	QRR	LUL	LUU	LDL	CHOL	UaL
No. of clock cycles	59	75	22	20	44	63	84
Max freq (MHz)	76.6	65.5	78.6	67.7	73.1	71.8	**210.06**
Computation time (us)	0.77	1.15	0.28	0.30	0.60	0.88	**0.40**

The Table V shows the comparison of computation time of parallel UaL factorization method of 4x4 matrix with the existing factorization techniques like LU, QR, LDL, Cholesky which are referred from [20]. If a parallelism of four is considered for the implementation of UaL decomposition then the resource utilization of UaL architecture will increase four times compared to sequential UaL but comparable to the other decomposition hardware implementations, and the computation time will decrease about 32% of sequential UaL computational time (see Table IV).

VI. CONCLUSION

The UaL decomposition method is computationally having higher efficiency and also less effected with round-off errors when compared with the regular LU decomposition method. The UaL decomposition algorithm itself shows high division operations are removed or made with zero hardware architecture corresponding to the UaL decomposition is proposed, and modeled using Verilog and simulated and synthesized by targeting FPGA Virtex-5 board using Xilinx Vivado.

As UaL decomposition exhibits natural parallelism similar to the LU decomposition in terms of processing steps its corresponding hardware implementation on FPGA not only as fast as

LU but also shows high computational efficiency and accuracy than LU. Therefore, the proposed UaL decomposition technique-based hardware architecture can be used in place of LU decomposition which can further improves the accuracy and computational efficiency of matrix decomposition. As the future work there is a chance to improve resource utilization and frequency even more by utilizing existing pipelined and parallelism techniques.

REFERENCES

1. M. K. Jaiswal and N. Chandrachoodan, "FPGA-Based High-Performance and Scalable Block LU Decomposition Architecture," in IEEE Transactions on Computers, vol. 61, no. 1, pp. 60-72, Jan. 2012, doi: 10.1109/TC.2011.24.
2. X. Ge, H. Zhu, F. Yang, L. Wang and X. Zeng, "Parallel sparse LU decomposition using FPGA with an efficient cache architecture," 2017 IEEE 12th International Conference on ASIC (ASICON), Guiyang, 2017, pp. 259-262, doi: 10.1109/ASICON.2017.8252462.
3. R. Hashemian, "UaL Decomposition, an Alternative to the LU Factorization of MNA Matrices," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 4, pp. 630-634, April 2020, doi: 10.1109/TCSII.2019.2924898.
4. Y. Wang, H. Tao, S. Xiao and H. Dai, "An implementation architecture design of LU decomposition in resource-limited system," 2015 IEEE International Symposium on Systems Engineering (ISSE), Rome, 2015, pp. 261-265, doi: 10.1109/SysEng.2015.7302767.
5. G. P. Kumar and C. Ramesh, "Implementation of an Area Efficient High Throughput Architecture for Sparse Matrix LU Factorization," 2019 3rd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2019, pp. 1-6, doi: 10.1109/IEMENTech48150.2019.8981319.
6. R. Wu and X. Xie, "Two-Stage Column Block Parallel LU Factorization Algorithm," in IEEE Access, vol. 8, pp. 2645-2655, 2020, doi: 10.1109/ACCESS.2019.2962355.
7. M. S. Feali, A. Ahmadi, A. Hamidi and M. Ahmadi, "Fixed-point arithmetic error analysis of sparse LU decomposition on FPGAs," 2017 International Symposium on Signals, Circuits and Systems (ISSCS), Iasi, 2017, pp. 1-4, doi: 10.1109/ISSCS.2017.8034900.
8. R. Mahfoudhi, "High Performance Recursive LU Factorization for Multicore Systems," 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, 2017, pp. 668-674, doi: 10.1109/AICCSA.2017.199.
9. K. F. K. Jiavana and N. Gurjar, "Stochastic multiplier and divider for stochastic LU decomposition," 2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2), Chennai, 2017, pp. 1-5, doi: 10.1109/ICNETS2.2017.8067884.
10. X. Ge, H. Zhu, F. Yang, L. Wang and X. Zeng, "Parallel sparse LU decomposition using FPGA with an efficient cache architecture," 2017 IEEE 12th International Conference on ASIC (ASICON), Guiyang, 2017, pp. 259-262, doi: 10.1109/ASICON.2017.8252462.
11. V. S. Rana, M. Lin and B. Chapman, "A Scalable Task Parallelism Approach for LU Decomposition with Multicore CPUs," 2016 Second International Workshop on Extreme

Scale Programming Models and Middlewar (ESPM2), Salt Lake City, UT, USA, 2016, pp. 17-23, doi: 10.1109/ESPM2.2016.008.

12. K. He, S. X. -. Tan, H. Wang and G. Shi, "GPU-Accelerated Parallel Sparse LU Factorization Method for Fast Circuit Analysis," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 3, pp. 1140-1150, March 2016, doi: 10.1109/TVLSI.2015.2421287.
13. Siddhartha and N. Kapre, "Heterogeneous dataflow architectures for FPGA-based sparse LU factorization," 2014 24th International Conference on Field Programmable Logic and Applications (FPL), Munich, Germany, 2014, pp. 1-4, doi: 10.1109/FPL.2014.6927401.
14. Siddhartha and N. Kapre, "Breaking Sequential Dependencies in FPGA-Based Sparse LU Factorization," 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, Boston, MA, USA, 2014, pp. 60-63, doi: 10.1109/FCCM.2014.26.
15. M. Eljammaly, Y. Hanafy, A. Wahdan and A. Bayoumi, "Hardware implementation of LU decomposition using dataflow architecture on FPGA," 2013 5th International Conference on Computer Science and Information Technology, Amman, Jordan, 2013, pp. 298-302, doi: 10.1109/CSIT.2013.6588795.
16. Baoguang Fang, Shuqiang Chen and Xulong Wei, "Single-precision LU decomposition based on FPGA compared with CPU," 2012 International Conference on Computational Problem-Solving (ICCP), Leshan, China, 2012, pp. 302-305, doi: 10.1109/ICCPS.2012.6384247.
17. G. Wu, X. Xie, Y. Dou, J. Sun, D. Wu and Y. Li, "Parallelizing sparse LU decomposition on FPGAs," 2012 International Conference on Field-Programmable Technology, Seoul, Korea (South), 2012, pp. 352-359, doi: 10.1109/FPT.2012.6412160.
18. G. Wu, Y. Dou, J. Sun and G. D. Peterson, "A High Performance and Memory Efficient LU Decomposer on FPGAs," in IEEE Transactions on Computers, vol. 61, no. 3, pp. 366-378, March 2012, doi: 10.1109/TC.2010.278.
19. Y. Shao, L. Jiang, Q. Zhao and Y. Wang, "High Performance and Parallel Model for LU Decomposition on FPGAs," 2009 Fourth International Conference on Frontier of Computer Science and Technology, Shanghai, China, 2009, pp. 75-79, doi: 10.1109/FCST.2009.66.
20. A. A. Hussain, N. Tayem and A. Soliman, "Matrix Decomposition Methods for Efficient Hardware Implementation of DOA Estimation Algorithms: A Performance Comparison," 2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE), 2019, pp. 1-7, doi: 10.1109/ICRAIE47735.2019.9037778.
21. T.L. Pillage, R.A. Rohrer, C. Visweswariah, Electronic Circuit & System Simulation Methods, McGraw-Hill, Inc., New York, 1995.
22. L.W. Nagel, "SPICE2, A computer program to simulate semiconductor circuits," Univ. of California, Berkeley, CA, Memorandum no. ERL-M520, 1975.
23. Serene Jose and Sonali Agrawal “Single precision Floating point divider design” in International Journal of Computational Engineering Research, May-June 2012.
24. Suresh,N.V,Satish,Kumar.P,"Design and implementation of fast floating point multiplier unit” 2015 International Conference on VLSI Systems, Architecture, Technology and Applications, VLSI SATA 2015, Institute of Electrical and Electronics Engineers Inc.

25. Aswini, R. Chinthala and N. S. Murty, "Area Efficient Architecture for high speed wide data adders in Xilinx FPGAs," 2019 International Conference on Computer Communication and Informatics (ICCCI), 2019, pp. 1-4, doi: 10.1109/ICCCI.2019.8822204.