# High Performance Reconfigurable Computing for Cholesky Decomposition

Depeng Yang, Gregory D. Peterson and Husheng Li

EECS Department, University of Tennessee at Knoxville, TN, 37996

*ABSTRACT*— **This paper proposes a hardware accelerator for Cholesky decomposition on FPGAs by designing a single triangular linear equation solver. Good performance is achieved by reordering the computation of Cholesky factorization algorithms and thus alleviating the data dependency. The dedicated hardware architecture for solving triangular linear equations is designed and implemented for different accuracy requirements using customized precisions. Compared to the software on the Intel Xeon quad core microprocessor, our design achieves a speedup of 7~13.**

## I. INTRODUCTION

Cholesky factorization is often the most expensive step in numerically solving a positive definite linear system of equations, such as in solving least square problems in signal processing [1]. Due to the inherently recursive computation process for Cholesky decomposition, it is very difficult to obtain acceleration by exploiting parallelism on FPGAs. The associated division and square root operations in traditional standard Cholesky decomposition represent difficulties because of long latency and data dependency [2].

By introducing an extra diagonal matrix into the standard Cholesky decomposition we propose that the Cholesky factorization can be realized by designing a single triangular linear equation solver implemented on FPGAs. This eliminates the division dependency and thus improves the data throughput as well as system performance. By exploiting parallelism and designing the dedicated process engine with deep pipelines on FPGAs, we can achieve a significant computation speedup.

## II. CHOLESKY DECOMPOSITION AND IMPLEMENTATION

The traditional Cholesky decomposition is associated with divisions, square root operations, and heavy data dependency. Thus it is difficult to achieve computation acceleration even with a deep pipeline. To alleviate data dependency, separate the division, and avoid square root operations [2][3][7], a diagonal matrix is introduced to avoid the square root, given by:

$$A = LDL^T \qquad (1)$$

where $D$ is a diagonal matrix. This makes $L$ become an atomic lower triangular matrix with all unit elements in the diagonal line.

In this paper, we introduce a novel method to calculate *LDL* Cholesky decompositions as in Eq.(1). Assume a symmetric *n-1* by *n-1* matrix $A_{n-1}$ has been factored as:

$$A_{n-1} = L_{n-1}D_{n-1}L_{n-1}^T \qquad (2)$$

And that based on matrix $A_{n-1}$ the $n$ by $n$ symmetric matrix $A_n$ is partitioned into a *2x2* block matrix which consists of the matrix $A_{n-1}$ in Eq.(2), a column vector $\boldsymbol{s}$ and a scalar number $t$, whose Cholesky decomposition is given by:

$$A_n = \begin{pmatrix} A_{n-1} & s \\ s^T & t \end{pmatrix} = \begin{pmatrix} L_{n-1} & 0 \\ m^T & 1 \end{pmatrix} \begin{pmatrix} D_{n-1} & 0 \\ 0 & d_n \end{pmatrix} \begin{pmatrix} L_{n-1}^T & m \\ 0 & 1 \end{pmatrix} \qquad (3)$$

where $\boldsymbol{s}$ and $\boldsymbol{m}$ are the *n-1* column vectors; $t$ and $d_n$ are scalars. Obviously, one only needs to calculate the column vector $\boldsymbol{m}$ and diagonal element $d_n$ based on calculations in Eq.(2). It is easy to verify that [3][5]:

$$L_{n-1}D_{n-1}m = s \qquad (4)$$

$$d_n = t - s^T D_{n-1} s = t - \sum_i^{n-1} s_i^2 d_i \qquad (5)$$

Initially from the most upper left element $A_1 = L_1 D_1 L_1^T$, where $L_1=1$ and $D_1=A_{11}$; we successively obtain decomposition $A_2 = L_2 D_2 L_2^T$, where $L_2$ and $D_2$ is based on previous results by solving the $\boldsymbol{m}$ and $d_n$ from Eq.(4) and Eq.(5). The matrices $L_3$, $D_3$,…, $L_{n-1}$, $D_{n-1}$, $L_n$, and $D_n$ are computed in sequence through updating the new column vector $\boldsymbol{m}$ and scales $d_n$ by solving Eq.(4) and Eq.(5).

Therefore, Cholesky decomposition of $A_n$ can be obtained by iteratively solving triangular linear equations starting with $A_1$. Compared with the complicated procedure for standard Cholesky decomposition [2], our method is associated with only one triangular linear equation solver. Using this approach a dedicated hardware solver for computation acceleration is designed and implemented on FPGAs.

## III. TRIANGULAR LINEAR EQUATIONS SOLVER

The key in our modified Cholesky decomposition is to obtain the new vector $\boldsymbol{m}$ by solving $\boldsymbol{Lz = s}$ and $\boldsymbol{Dm = z}$, equivalently solving Eq.(4), which is expanded to:

$$
\begin{aligned}
z_1 &= s_1 / l_{11} \\
z_2 &= (s_2 - l_{21} z_1) / l_{22} \\
z_3 &= (s_3 - l_{31} z_1 - l_{32} z_2) / l_{33} \\
&\vdots \\
z_n &= (s_n - \sum_{i}^{n-1} l_{ni} z_i) / l_{nn}
\end{aligned}
\qquad (6)
$$

$$
m_n = z_n / d_n \qquad (7)
$$

where $d_n$ is the diagonal element in the diagonal matrix $D$; $\boldsymbol{z}=\{z_1,z_2,\ldots z_n\}$, $\boldsymbol{s}=\{s_1,s_2,\ldots s_n\}$, $l_{11}=l_{22}=l_{33}=\ldots=l_{nn}=1$, and $\boldsymbol{m}=\{m_1,m_2,\ldots m_n\}$. Note that in solving normal triangular linear equations where diagonal elements are not unit valued, the next step computation has to wait for previous division results fed back as inputs. However, the long latency of division, particularly when operands are in floating point [4], will greatly hurt system performance and make it very difficult to exploit computation parallelism. To circumvent this problem a diagonal matrix is introduced to make diagonal elements in the lower triangular matrix unit valued, which separates the division operation out of the pipeline and also avoids the square root operation [3][7], thus leading to improved parallelism and system throughput.

To exploit the parallelism, we demonstrate the time order of the computation for calculating Eq.(5) and Eq.(6) in Table 1. The current result $z_i$ will be fed back as the input for the next calculation. For instance, initially $z_1$ ($z_1 = s_1$) is provided as input for calculating $z_2$ in first step. After being computed, $z_2$ is then fed back for the $z_3$. Therefore the time delay between two steps determined by the hardware latency will decide computation speed. And if $\{l_{11}=l_{22}=l_{33}=\ldots=l_{nn}=1\}$ are not unit valued and division is needed, the long latency of the division will greatly hurt the performance. The benefits of introducing a diagonal matrix is to avoid long latency by separating divisions, which is implemented as an individual divider for solving Eq.(6).

The corresponding pipelined hardware architecture is illustrated in Fig 1. This triangular linear equation solver consists of multiple Process Elements (PEs) for computing $z_n$, result control logic for feeding back $z_n$, an individual divider to obtain $m_n$ , and a BRAM module for storing final solutions. Assume there are $n$ PEs so we need just one clock for computing $z_1, z_2, \ldots z_n$ in each step as shown in Table 1. In this case, we do not need FIFO1 and FIFO2 any more. In practice, when the matrix size is large we can create $m$ PEs where $m<<n$. Therefore, to compute $z_1$, $z_2$, $\ldots z_n$ in the first step we need at least $n/m$ clock cycles by multiplexing $m$ PEs. In such a way FIFO1 inside the PE is used to align the computation. This is because the computation of $z_1$, $z_2$, $\ldots$ $z_m$, are in $PE_1$, $PE_2$, $PE_m$, respectively, and computation of $z_{m+1}, z_{m+2}, \ldots z_{2m}$, are also in those PEs next clock. Similarly FIFO2 outside the PE is implemented to feed back $z_j$ as inputs for all PEs. The depth of FIFO1 and FIFO2 is adjusted according to the ratio $n/m$. Note that Cholesky decomposition needs to solve triangular equations several times. Multiple hardware solvers can operate in pipeline mode for best performance. Due to limited space, we only implement and test one solver.

The system performance is determined by the maximum achievable frequency and latency of the PEs, as well as the memory interface frequency and bandwidth. For a small size matrix, elements can be wholly stored inside FPGAs, and performance is dominated by the achievable frequency. But for large size matrices, a memory interface will become the bottleneck. Additionally the control logic which handles the whole computation process [6] can be realized by embedded microcontroller or host CPU.

Computation accuracy is determined by the precision. High accuracy requirements imply high precisions leading to higher memory bandwidth, and more hardware resources, but lower system frequency. There is a balance between computation speed and accuracy using different precisions with customized mantissa bits. Fig 2 depicts the log of the root mean square (RMS) error, defined in Eq.(8), for result elements of matrix $L$ and $D$ using double precision (s52e11) as a reference compared with other customized precisions by changing the mantissa from 20 to 51 bits in the computation .

$$
err = 10 \log \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} (\tilde{L}_{ij} - L_{ij})^2 / N^2 + \sum_{i=1}^{N} (\tilde{D}_i - D_i)^2 / N} \qquad (8)
$$

We test 1000 matrices for Cholesky decomposition with elements randomly distributed and with different mantissa size. Although the error is affected by the condition number of the original matrix $A$, the accuracy is exponentially improved with increasing mantissa size. Taking advantage of customized precision on FPGAs, a designer could balance the accuracy, precision, and hardware resources according to application requirements.

## IV. Results

We implemented our design in Fig.1 by using Xilinx ISE 9.2i floating IP cores, which follow the IEEE 754 standard but can be customized for solving triangular linear equations, $LD\boldsymbol{m}=\boldsymbol{s}$, using Eq.(6) and Eq.(7). Results are summarized in Table 2 for the hardware resource and performance of the implemented solver with 16 PEs on a Xilinx XC5VSX95T-2 FPGA (containing 14720 slices and 640 DSP48 modules). We use our hardware design to compare with software written in C and the same computation procedure on a microprocessor for solving 256x256 lower triangular linear equations. The CPU is a quad core 3GHz Intel Xeon X5450 with 6144KB cache and 2GB memory.

Note that 's20e8' represents customized floating point with 8 exponential bits and 20 bits signed mantissa, and so on for other customized precisions. We choose five different mantissa bits to test maximum achievable frequencies of the hardware solver. For large matrices, the memory interface determines how fast the data would be fed into the PEs and may become a bottleneck. For our application we suppose data communication frequency is set to 200MHz and 100MHz for single and double precision data transfers. We implemented the solver containing 16PEs, each with 16 clocks latency. The testing matrix size is 256x256. 'FPGA time' is the time of solving the triangular linear equations using forward substitution based on the memory interface frequency. We also utilized DSP48 modules for high performance. The 'speed up' represents the ratio of the time consumed on CPU ('CPU time') for solving same triangular linear equations to the time on FPGA ('FPGA time'). Also note that in Table 2 it only shows the speedup for a single hardware solver. When implementing multiple solvers for parallel processing several matrices, the speedup will be improved significantly.

## V. Conclusions

We proposed an innovative hardware implementation for realizing modified Cholesky decomposition by designing a hardware triangular linear equation solver. The deep pipelined hardware design has achieved a significant speed up compared with Intel Xeon quad core at 3GHz microprocessor. Future work includes implementing our Cholesky decomposition algorithms on GPGPUs and comparing with FPGA performance.

## References


[1] D. Yang, H. Li, G. Peterson and A. Fathy, "Compressed sensing based UWB receiver: Hardware compressing and FPGA reconstruction," *Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, Mar. 2009

[2] O. Maslennikow, V. Lepekha, A. Sergiyenko, A. Tomas3, and R. Wyrzykowsk, "Parallel Implementation of Cholesk LLT-Algorithm in FPGA-Based Processor," *Springer Verlag Berlin Heidelberg*, pp. 137-147, 2008

[3] Wikipedia, "Cholesky Decomposition", *wiki encyclopedia* http://en.wikipedia.org/wiki/Cholesky_decomposition

[4] "Pipelined Divider v03", http://www.xilinx.com, Jan. 2006

[5] Manuel Loth, "Sequential Least Squares: Cholesky update" http://sequel.futurs.inria.fr/loth/seq_ls.pdf, Dec., 2007

[6] J. Sun, G. Peterson, and O. Storaasli, "High Performance Mixed-Precision Linear Solver for FPGAs," *IEEE Transaction on Computers*, vol. 57, no. 12, Dec. 2008.

[7] S. Bellis, W. Marnane, and P. Fish, "Alternative Systolic Array for Non-square-root Cholesky Decomposition," *Computers and Digital Techniques,* vol. 144, Mar. 1997.


Table 1. Computation order for triangular linear equations

| Step 1 | Step 2 | … | Step n-1 |
|---|---|---|---|
| $(z_1=s_1)$<br>$z_2=s_1-l_2z_1$<br>$z_3=l_3z_1$<br>$z_4=l_4z_1$<br>……<br>$z_n=l_nz_1$ | $z_3=s_3-l_3z_2-z_3$<br>$z_4=-l_4z_2-z_4$<br>……<br>$z_n=-l_nz_2-z_n$ | .<br>.<br>.<br>. | $z_n=s_n-l_{n-1n-1}z_{n-1}-z_n$ |
| $m_1=z_1/d_1$ | $m_2=z_2/d_2$ | . | $m_n=z_n/d_n$ |
| Feed back $z_2$ | Feed back $z_3$ | . | |

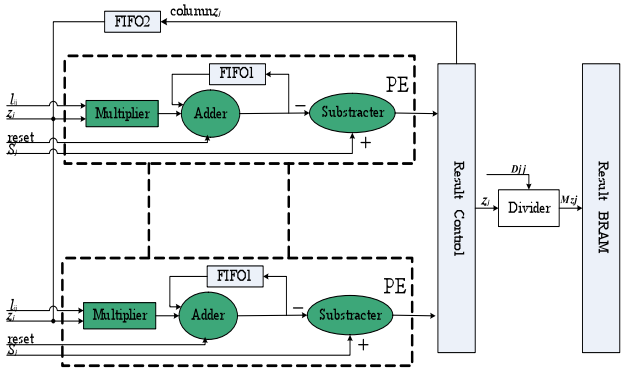


Fig 1. Architecture for solving triangular linear system

| Design | s20e8 | s23e8 (single) | s32e11 | s46e11 | s52e11 (double) |
|---|---|---|---|---|---|
| Freq (MHz) | 265 | 255 | 220 | 206 | 175 |
| Slices | 19% | 24% | 34% | 62% | 73% |
| DSP48 | 7% | 22% | 24% | 35% | 47% |
| CPU | 140.4875us | | | 145.9826us | |
| Interface Freq | 200MHz | | | 100MHz | |
| FPGA | 10.96 us | | | 21.76 us | |
| speedup | ~13 | | | ~7 | |

Table 2. Performance and hardware resources on the FPGA

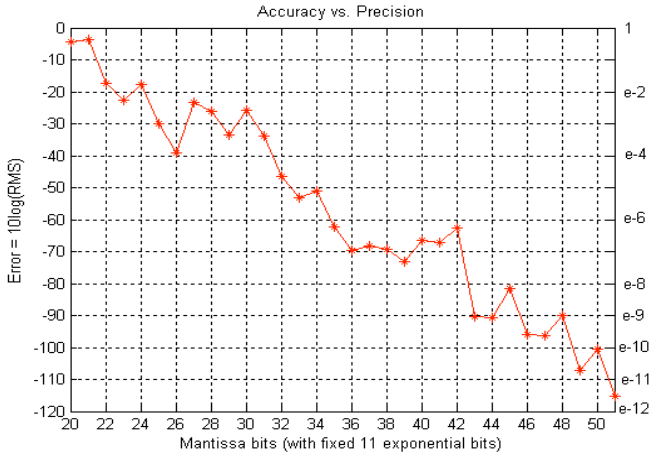


Fig 2. Accuracy using customized mantissa bits