International Journal of Computational Science and Engineering

ISSN online: 1742-7193 - ISSN print: 1742-7185
https://www.inderscience.com/ijcse

Investigation on the optimisation of Cholesky decomposition algorithm based on SIMD-DSP

Huixiang Li, Huifu Zhang, Anxing Xie, Yonghua Hu, Wei Liang

DOI: 10.1504/IJCSE.2024.10061859

Article History:

Received:	17 March 2022
Last revised:	17 July 2022
Accepted:	18 July 2022
Published online:	25 January 2024

Investigation on the optimisation of Cholesky decomposition algorithm based on SIMD-DSP

Huixiang Li, Huifu Zhang, Anxing Xie, Yonghua Hu* and Wei Liang

Hunan Key Laboratory for Service computing and Novel Software Technology,
School of Computer Science and Engineering,
Hunan University of Science and Technology,
Xiangtan, China
Email: 1449488105@qq.com
Email: hfzhang@hnust.edu.cn
Email: axie@mail.hnust.edu.cn
Email: huyh@hnust.cn
Email: wliang@hnust.edu.cn
*Corresponding author

Abstract: With the development of high-performance SIMD-DSP processors, corresponding highly efficient algorithms for matrix decomposition play an important role in the hardware performance of such processors. Cholesky decomposition is a fast decomposition method for symmetric positive definite matrices, which is widely used in matrix inversion and linear equation solving. According to the hardware characteristics of the FT-M7002 processors, in this paper, we optimise the algorithm in several ways. If hardware has on-chip double-buffered memory, the parallel process of DMA transmitting and calculating is specially designed, which can hide most of the time cost of data movement and further improve the algorithm's performance. The experimental results based on the FT-M7002 processor show that the performance of the optimised algorithm is 3.8~5.64 times that of the serial algorithm, and 1.39~2.14 times that of the TI library function.

Keywords: Cholesky decomposition; digital signal processor; DSP; single instruction multiple data; SIMD.

Reference to this paper should be made as follows: Li, H., Zhang, H., Xie, A., Hu, Y. and Liang, W. (2024) 'Investigation on the optimisation of Cholesky decomposition algorithm based on SIMD-DSP', *Int. J. Computational Science and Engineering*, Vol. 27, No. 1, pp.28–35.

Biographical notes: Huixiang Li is a Master's student at the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interest is algorithmic parallel optimisation.

Huifu Zhang is a Professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interest is embedded system application.

Anxing Xie is a Master's student at the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interest is compilation technology.

Yonghua Hu is a Professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interest is compilation technology and parallel computing.

Wei Liang is a Professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interest is trusted computing systems and integrated circuit copyright protection.

1 Introduction

Digital signal processor (DSP) is widely used in the fields of communication, radar, voice recognition, image processing, etc. This benefits from its simple structure, low power consumption, low price and easy programmability. With the continuous development of large-scale integrated circuits and information technology, the architecture of DSP has also undergone significant changes. For high-performance DSPs, the technologies single instruction multiple data (SIMD), very long instruction word (VLIW), multi-core and efficient storage technologies are mainly

used to provide high parallelism (Wang et al., 2011; Yu, 1996). In embedded systems, signal processing applications with a large number of computing requirements have put forward an urgent requirement for high-performance computing (HPC), which drives the rapid development of high-performance DSP processors with SIMD architecture (Robelly et al., 2008). In the SIMD architecture processor, multiple parallel units share a set of logic for fetching, decoding, and dispatching. The control structure is simple, which can effectively reduce the hardware cost and achieve higher performance at lower power consumption. Thus, SIMD-DSP is favoured by the HPC field, and its lower cost and higher performance-to-power ratio can play a certain advantage in mobile computing platforms (Yang et al., 2021).

In recent years, many commercial processors include some SIMD extension components and corresponding instruction sets, such as Intel AVX instruction set (Adriaansen et al., 2016), and ARM's ENON instruction set (Henrichs et al., 2009). Famous SIMD-DSP processors include SODA (Lin et al., 2007) and AnySP (Woh et al., 2009) proposed by Michigan University, Maven (Lee et al., 2016) proposed by MIT University, BBE64 (Rowen et al., 2011) proposed by Tensilica, FT-M7002 (Wang et al., 2021) proposed by the National University of Defense Technology, etc. As the continuous improvement of hardware brings performance improvement, it also puts forward higher requirements for the algorithm optimisation of the upper-layer software library. Therefore, it is of great significance to optimise the corresponding algorithm program according to the hardware characteristics and give full play to the hardware architecture advantages of the high-performance SIMD-DSP processor.

Solving large-scale linear equation sets is an important problem in the HPC field, and the matrix's triangular decomposition is often used to simplify the solution process. The Cholesky decomposition is an important method for trigonometric decomposition of positive definite symmetric matrices. When compared with general LU decomposition, its calculated amount is reduced by about half. Thus, researchers have conducted extensive applied research on Cholesky decomposition, both from the application and optimisation aspects. The Cholesky decomposition algorithm plays a critical role in many fields, such as image processing (Xu et al., 2019), web service computing (Tang et al., 2021), and other applications of matrix operation.

In the research on the application of the Cholesky decomposition algorithm, Gao and Li (2015) proposed an efficient parallel preprocessing conjugate gradient algorithm, which optimises vector operations by dividing multiple vector operations into multiple groups into a single kernel, overcoming the disadvantage of the forward and backward substitution that it is difficult to parallelise on the GPU, and used Cholesky decomposition to solve large sparse linear systems caused by numerical solutions of 3D parabolic equations. To solve the problem that the traditional extreme learning machine cannot overcome the problem of sample noise and imbalance, Jin (2019) used the KFCM algorithm combined with the proportion of the samples to obtain the sample weight matrix, and used the Cholesky decomposition to invert the matrix to speed up the training process. Herholz and Sorkine-Hornung (2020) proposed a novel linear solver for interactive parametric tasks, enabling a seamless and interactive workflow by updating the Cholesky decomposition of linear systems to reflect the new boundary conditions. Nino-Ruiz et al. (2015) presented an efficient parallel implementation of a Kalman filter based on Cholesky decomposition, decomposing a domain into multiple subdomains, and eliminating inter-processor communication in subdomain computations.

In the research on the optimisation of the Cholesky decomposition algorithm, Dorris et al. (2018) modified the tiled Cholesky decomposition of PLASMA, used OpenMP tasks as the scheduling mechanism, and introduced the algorithm optimisation in the Intel Xeon Phi architecture processor in detail. Haidar et al. (2017) developed efficient code executed entirely by GPU, addressing the tuning challenges associated with the communication between CPU and GPU. Lemaitre et al. (2018) proposed a code generator that directly generates SIMD codes for the Cholesky decomposition of small matrices and Kalman filters. Liu et al. (2016) analysed the data dependencies of Cholesky decomposition based on the field programmable gate array (FPGA) and studied the fine-grained pipeline parallel structure and implementation of Cholesky decomposition. Shen and Dai (2019) and others optimised the Cholesky decomposition algorithm by improving the utilisation of local memory in the heterogeneous system of CPU + GPU.

Although the above kind of literature has achieved good results in the optimisation of Cholesky decomposition, they have ignored the optimal implementation on SIMD-DSPs, and the existing optimisation methods cannot adapt to the structure of high-performance SIMD-DSPs. In this paper, we first utilised the symmetry of a positive definite matrix, and used the upper triangular matrix calculation instead of the lower triangular matrix calculation to obtain the transposed result matrix L^T to avoid discontinuous access to the memory. Then, we build a direct memory access (DMA) double-buffered transmitting model, and hide the latency between computation and data movement in the calculation process. Besides, we use the vector shuffling unit to realise the effect of broadcasting a single value as a vector. We also use generalised optimisation methods such as loop unrolling and software pipelining to exploit the instruction-level parallelism of the algorithm fully. We evaluate our optimised algorithm of Cholesky decomposition based on vector SIMD-DSP, and demonstrate the significant performance improvement for the corresponding serial algorithm and the corresponding library functions of the TI.

2 Basic principles of Cholesky decomposition algorithm

The Cholesky decomposition is defined as for a positive definite symmetric matrix $A = [a_{ij}]$, there is a unique lower

triangular matrix L whose main diagonal elements are positive, and L satisfies

$$A = LL^T \tag{1}$$

where L^T is the transpose matrix of L. According to equation (1), a positive definite symmetric matrix A can be written as (Golub and Loan, 1986)

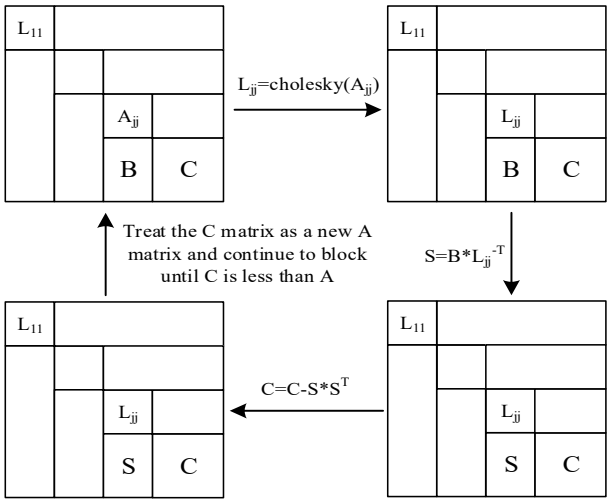
$$A = \begin{bmatrix} A_{11} & B^T \\ B & C \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ S & \overline{L} \end{bmatrix} \begin{bmatrix} L_{11}^T & S^T \\ 0 & \overline{L}^T \end{bmatrix} \tag{2}$$

and according to the characteristics of partitioned matrices multiplication, we have

$$\begin{aligned} L_{11} &= cholesky(A_{11}) \\ S &= B \times L_{11}^{-T} \\ \overline{L} \times \overline{L}^T &= C - S \times S^T \end{aligned} \tag{3}$$

For the submatrix C in equations (2) and (3), it can be viewed as a new input matrix, and we can continue to partition it. This process can be done recursively according to equation (3) until that C is smaller than A_{11} in size. Then, the final result matrix will be obtained. It is a partitioning solution method, and the calculation process of the Cholesky decomposition of the partitioned matrix is shown in Figure 1. As seen in Figure 1, the size of new matrix A will decrease with decomposition in the Cholesky decomposition algorithm.

Figure 1 Cholesky decomposition flowchart



In the calculation process, the access of data in the on-chip data cache is much faster than that of data in an independent memory chip, so we need to transmit the data in memory to the on-chip data cache before calculating them. However, for the large matrix case, it may not be possible to put the entire input matrix into the processor's on-chip cache at the beginning of Cholesky decomposition. If the on-chip cache can double-buffering, we can develop an optimisation for the data transmitting, which allows the transmitting of some data to be parallelised with the calculation of other data. This parallel processing can continue until A can be placed completely in the on-chip data cache.

3 Implementation and optimisation of Cholesky decomposition based on vector SIMD-DSP

3.1 Generation method of upper triangular matrix based on Cholesky decomposition according to vector memory access characteristics

For a vector SIMD processor, its vector length is fixed. Thus, if the size of a matrix is inconsistent with the vector length during data processing, we will face the data misalignment problem when we implement the Cholesky decomposition algorithm for the vector SIMD processor. During the calculation process, the number of data to calculate (i.e., calculation width) being smaller than the vector length will occur in the Cholesky decomposition of matrix A_{jj}. It is the so-called short vector problem. The handling of the short vector is a considerable part of the whole work of the Cholesky decomposition for submatrix, and it wastes some computing power of the processor. Therefore, we set the A_{jj} in Figure 1 as a diagonal element to implement the Cholesky decomposition algorithm for the vector SIMD processor. Then, the formula

$$L_{jj} = cholesky(A_{jj}) \tag{4}$$

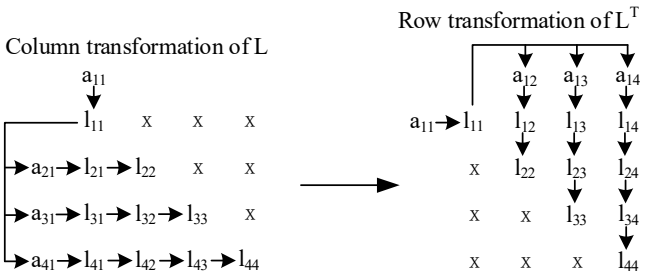
can be replaced by

$$l_{jj} = \sqrt{a_{jj}}. \tag{5}$$

In the generation algorithm for the lower triangular matrix of Cholesky decomposition, the updating of columns will discontinuously access corresponding memory units. The efficiency of this discontinuous access is very low, only 1/15 of the efficiency of continuous access (Cupertino et al., 2010). It will greatly influence the algorithm's performance when the on-chip cache of a vector SIMD processor can be accessed parallelly by the processing elements of SIMD unit. Therefore, to map the generation algorithm to a vector SIMD processor, it is necessary to convert discontinuous memory access to continuous access.

Considering the symmetric geometric characteristics of matrix A, the upper triangular matrix L^T can be generated according to the same calculation principle to replace the lower triangular matrix L. When using L, it is only necessary to replace the access of the element $L[i, j]$ of the lower triangular matrix with the access of the element $L^T[j, i]$ of the upper triangular matrix. By this conversion, the column updating in the calculation process of generating the lower triangular matrix is changed into the row updating in generating the upper triangular matrix. Because the data in a row are continuously stored in the cache, the algorithm can calculate them parallelly. The process of calculating the lower triangular matrix and that of calculating the upper triangular matrix is shown in Figure 2, where 'x' represents irrelevant matrix elements.

Figure 2 Schematic diagram of generating upper triangular matrix and lower triangular matrix



In the process of generating the matrix L^T, A will be updated several times (which is decided by the row number of A). As seen in Figure 2, a cycle of updating A can be divided into the following steps:

1 updating the corresponding diagonal element

2 updating other elements in the row where the diagonal elements are located

3 updating the corresponding submatrix.

Section 3.2 will introduce specific optimisation methods for these three steps.

3.2 Optimisations for Cholesky decomposition

3.2.1 Reusing the data of diagonal element calculation

The method for calculating the square root of the diagonal elements is: firstly, the vector process element (VPE) loads the diagonal element a_{jj} of the first row of the current A. Then, we calculate the reciprocal square root l_{jj}^{-1} of a_{jj}, and then we use the reciprocal of l_{jj}^{-1} to get the new value of diagonal element l_{jj}. In order to ensure the precision of the reciprocal square root, the Newton-Rhason iteration formula can be used. The iterative formula to calculate the reciprocal square root is shown in equation (6), where $X[n]$ is the initial reciprocal square root of v, and $X[n + 1]$ is the result after iteration. The iterative formula to calculate the reciprocal is shown in equation (7), where $X[n]$ is the initial reciprocal of v, and $X[n + 1]$ is the result after the iteration. For each iterative formula, the precision will be doubled if used once.

$$X[n+1] = X[n]\left(1.5 - (v/2) \times X[n] \times X[n]\right) \tag{6}$$

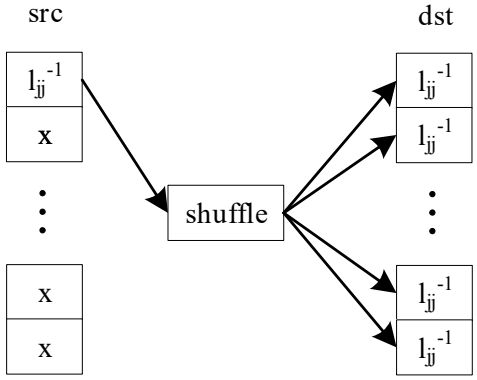
$$X[n+1] = X[n]\left(2 - v \times X[n]\right) \tag{7}$$

The intermediate data l_{jj}^{-1} obtained in the above calculation process can be preserved in the register for the update of other elements in the first row in Section 3.2.2, which allows converting the corresponding division into multiplication there.

3.2.2 Vectorisation of updating the rows of matrix

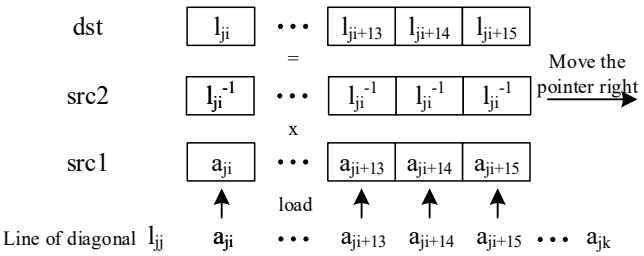
The shuffling unit is generally equipped in a SIMD processor to implement copying data at the register level in the vector processing unit. For certain source vectors, by configuring a specific shuffling mode, we can obtain a new vector from different source vectors. The value of the elements in the result vector can be from any element of other vectors. In the update process of other elements in the row where the diagonal element of the current A is located, the l_{jj}^{-1} calculated in Section 3.2.1 is copied to all VPEs for subsequent calculations by shuffling. This process is shown in Figure 3.

Figure 3 Broadcast flowchart of a single element in a vector



For updating other elements in the row corresponding to the diagonal element, other elements except for the diagonal element in this row will be loaded into vectors and multiplied by l_{jj}^{-1}. This calculation process is shown in Figure 4, where dst = src1 ∗ src2, and dst, src1 and src2 are vectors which can be processed in SIMD unit. Moreover, further data-level parallelism can be exploited through loop optimisation methods such as loop unrolling and software pipelining.

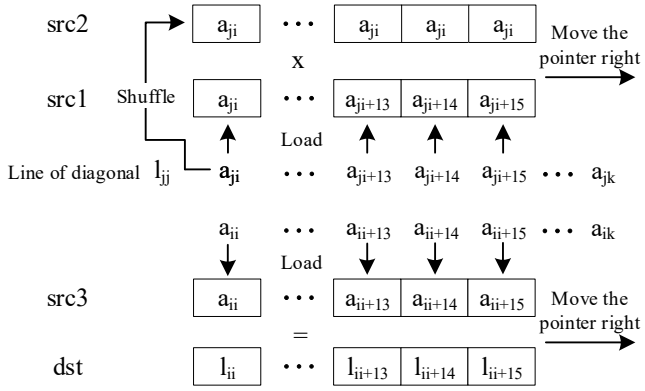
Figure 4 The updating flowchart of the row corresponding to a diagonal element



3.2.3 Vectorisation of updating submatrix

The updating of the submatrix corresponding to a diagonal element can be divided into a series of sub-processes, each of which updates a row of the matrix. Assuming that the first row of the current A is a part of the row j of the whole matrix, and the row of the currently updated submatrix is a part of row i of the whole matrix, the updating process of a row of the submatrix is shown in Figure 5.

Figure 5 Process of updating submatrix



The update of a row in the submatrix in Figure 5 can be described as the following steps:

1 Taking out the a_{ji} element and expanding it with the shuffling method in Figure 3 to obtain src2.

2 Taking out all the elements of the j^{th} row in turn to obtain src1.

3 Taking out all the elements of row i in turn to get src3.

4 Calculating the result values by formula dst = src3 – src2 ∗ src1, and then writing dst into its corresponding position in the matrix.

3.2.4 Parallel optimisation of DMA transmission and computation for large matrix

DMA provides a high-speed data transfer path, enabling access to specific memory resources according to pre-configured parameters. For the Cholesky decomposition of large matrices, if the on-chip cache of the processor cannot accommodate the entire input matrix, the decomposition process must exchange data between the cache and the off-chip memory multiple times. Assuming that the on-chip cache has a 'ping-pong' storage function, a DMA double-buffered transmitting model can be constructed to realise the parallelisation of the two processes of calculation and DMA transmitting, reducing the special time overhead of data movement in the calculation process. The DMA double-buffered transmitting model is as follows: the on-chip cache is divided into two parts, such as buffer0 and buffer1; for each part, the space is used to store the first row of the current A and a part of the corresponding submatrix C. The steps of the whole calculation process are as follows:

1 Transmit the first row of the current A into the corresponding areas in buffer0 and buffer1 by DMA.

2 Computes the diagonal elements stored in buffer0 and buffer1.

3 Update elements in the first row of the current A, except the diagonal element.

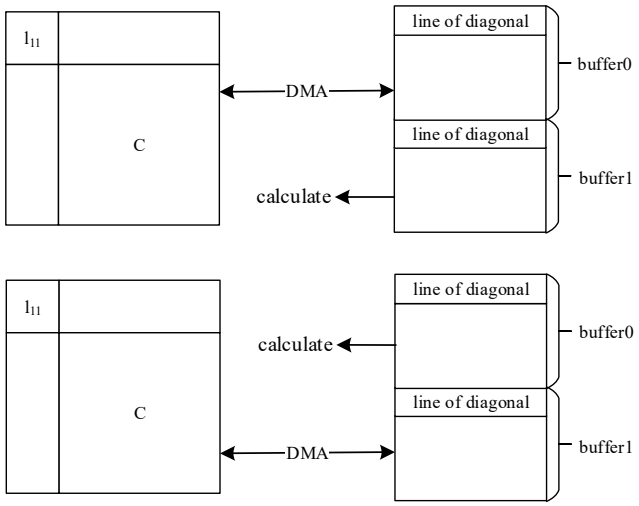
4 Transmit the content of the corresponding submatrix C part by part into the buffer0 or the buffer1 in turn for calculation.

5 Consider the matrix C as a new matrix A, and judge whether A can be put into the on-chip cache. If not, skip to Step 1. Otherwise, continue to execute the next step.

6 Transmit all the data of A to the on-chip cache for calculation.

The parallel processing principle of DMA transmission and calculation for updating the submatrix is shown in Figure 6.

Figure 6 Matrix update calculation and DMA parallel graph



4 Experimental result analysis

This section describes the performance improvement of the Cholesky decomposition optimisation algorithm over the standard Cholesky decomposition serial algorithm on the FT-M7002 processor. Besides, to reflect the performance of the optimised Cholesky decomposition algorithm for the FT-M7002 processor, the library function DSPF_sp_cholesky algorithm of TI's TMS320C6678 processor was introduced as a comparison.

4.1 Experimental environment

4.1.1 FT-M7002 processor

FT-M7002 is a 40 nm high-performance DSP processor independently developed by the National University of Defense Technology. It integrates 1 RISC CPU core and 2 FT-MT2 DSP cores on the chip, and the entire processor uses a three-level storage structure. A single DSP core has 32 kb scalar storage space scalar memory (SM) and 512 kb vector storage space vector memory (VM). When running at 1 GHZ, the peak double-precision floating-point performance is up to 100 GFLOPS, and the peak single-precision floating-point performance is up to 200 GFLOPS (Wang et al., 2021).

The DSP core of the FT-M7002 processor is based on the VLIW structure, including a scalar processor unit SPU (scalar process unit) and a vector processing unit VPU (vector process unit). These two processing units are tightly

coupled (Chen et al., 2021). The instruction dispatching component can provide instructions for SPU and VPU simultaneously. The SPU mainly performs serial instruction execution and flow control of the entire program. The VPU consists of 16 vector computing engines (VPE), which support up to 16 channels of 32 bit data. The VPU performs vector operations and provides parallel processing mainly for intensive calculations. The DMA can do fast data exchange between out-of-core DDR and SM or VM. When the memory access operations do not conflict, the VM can support two vectors read/writes simultaneously and two DMA read/writes requests. DMA transmitting and VM access can be implemented in parallel through reasonable data arrangement.

4.1.2 TMS320C6678 processor

TMS320C6678 is an eight-core processor released by TI in November 2010. Each core has both fixed-point and floating-point computing capabilities. It has 32 KB level 1 data (L1D) SRAM, 32 KB level 1 program (L1P) SRAM, and 512 KB local level 2 (LL2) SRAM. At the core frequency of 1.25 GHz, the fixed-point computing capability of a single core can reach 40 GMAC/s, and the floating-point computing capability of a single core can reach 20 GFLOPS/s (Zhou et al., 2014). It has been widely used in sonar, radar, communications, and other fields that require high fixed-floating-point computing capability and real-time performance.

TI has efficiently implemented library functions for its DSP series chips, including general and some special fields. TI has done a lot of algorithm optimisation according to its algorithm characteristics and chip structure characteristics and maximises the execution efficiency of library functions through the manual assembly. DSPF_sp_cholesky is a function in TI's DSP library. Therefore, the execution efficiency of DSPF_sp_cholesky represents the highest execution efficiency of the Cholesky decomposition algorithm on this kind of DSP processor. It is more convincing to compare the performance of the optimisation algorithm in this paper with it.

4.1.3 Testing method

For the Cholesky decomposition optimisation algorithm and the Cholesky decomposition serial algorithm implemented on the FT-M7002 processor, the parallelism of code is ensured by assembly encoding. To get the execution cycle for the algorithm, related timing functions for the on-chip timer function are called. The performance of the TMS320C6678 processor is measured by calling the corresponding library function DSPF_sp_cholesky in the CCS5.5.0 simulator, and the execution cycle number of this library function is measured by calling the timing function. The parameters of the experimental platform are shown in Table 1.

Table 1 Experimental platform parameters

Platform	*FT-M7002*	*TMS320C6678*
Frequency / GHz	1.0	1.25
LID / KB	32	32
AM / KB	512	-
Software platform	FT-M7002 IDE	CCS5.5.0

4.2 Performance analysis

For the small matrix case, the whole matrix can be transmitted into the vector buffer VM at one time in the FT-M7002 processor, and there is no need for DMA transmission and parallel computing processing during the processing. In this paper, the Cholesky decomposition is analysed experimentally for small matrices of order 128, order 160, order 192, order 228 and order 256. For these small matrix sizes, the corresponding cycle numbers obtained from the tests are shown in Table 2.

Table 2 Cycle number of the Cholesky decomposition algorithm in the mall matrix case

Matrix size	*Serial algorithm*	*Optimised algorithm*	*TI library functions*
128	3,621,520	977,296	1,358,861
160	6,807,840	1,595,968	2,304,982
192	11,463,920	2,400,688	4,274,809
224	17,855,808	3,403,616	6,759,964
256	26,316,144	4,659,808	9,989,944

Table 2 shows the acceleration ratios of the optimised Cholesky decomposition algorithm relative to the TI library function and the basic serial algorithm shown in Figure 7. It can be seen from Figure 7 that, as the matrix size increases, the speedup ratio of the optimised Cholesky decomposition algorithm relative to the serial algorithm and that to the TI library function increases steadily. In the above five groups of small-scale matrices, the speedup ratio of the optimisation algorithm relative to the basic serial algorithm is 3.8~5.64, and the speedup ratio of the optimisation algorithm relative to the TI library function is 1.39~2.14. It shows that the speedup effect by using the optimised Cholesky decomposition algorithm of this paper is obvious.

For the large matrix case, Figure 8 shows the acceleration ratio of Cholesky decomposition optimisation algorithm relative to the corresponding serial algorithm under the matrix sizes of 512th order, 768th order, 1,024th order, 1,280th order and 1,536th order.

As seen from Figure 8, as the size of the matrix increases, the advantage of the Cholesky decomposition algorithm that combines the optimisation of vector parallel computing and the optimisation of DMA double-buffered transmitting becomes greater. For the five input matrix sizes shown in Figure 8, a speedup of 7.8 to 14.1 times is achieved.

Figure 7 Acceleration ratio of Cholesky decomposition in the small matrix case

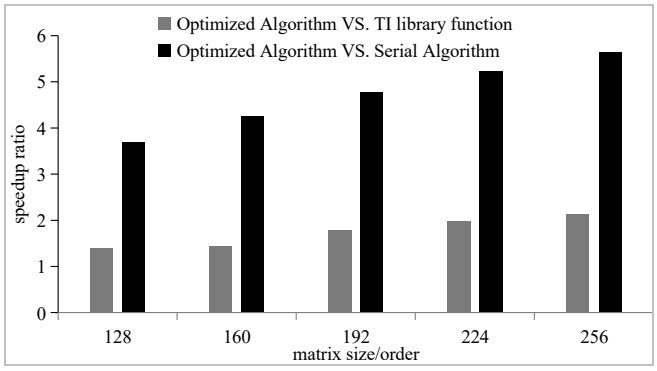
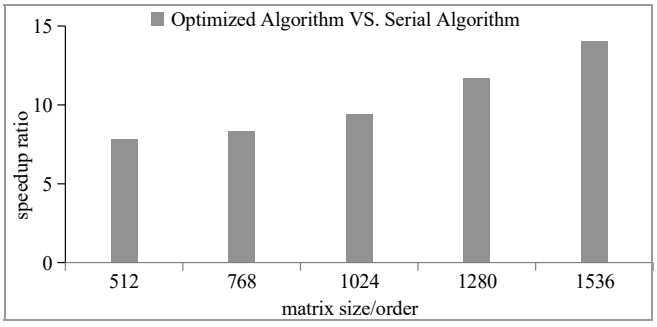


Figure 8 Acceleration ratio of Cholesky decomposition optimisation algorithm to serial algorithm in the large matrix case



For the accuracy of the decomposition result, we note that corresponding calculating instructions of architecture mainly decide this, and we just focus on the instruction-level parallelisation of the algorithm. For the FT-M7002 processor, we take the case that the input matrix order is 128 as an example. The average relative error between TMS320C6678 and FT-M7002 is 4.6749×10^{-5}. It demonstrates that our method is also effective and correct.

This section uses small-scale and large-scale matrices to demonstrate the effect of the optimised Cholesky decomposition algorithm on the FT-M7002 processor. The corresponding TI library function is introduced for comparative analysis. The experimental results show that the algorithm proposed in this paper has better performance. The performance improvement reasons are mainly as follows: firstly, we utilised the vector processing unit to calculate the data in parallel. This unit in the FT-M7002 processor has 16 VPEs. Besides, optimisation methods such as loop unrolling and software pipelining are used in our code. Secondly, we utilised the symmetry of the input positive definite matrix, and used the upper triangular matrix calculation instead of the lower triangular matrix calculation to obtain the result matrix L^T. It allows the algorithm to avoid discontinuous access to the memory. Thirdly, taking advantage of the double-buffered mechanism of the VM cache in the FT-M7002 processor, we realised the parallel processing of the two processes of calculation and data transmitting. In a word, the optimised method in this paper can take advantage of the architecture characteristics of vector DSP and brings good instruction-level parallelism to the algorithm.

5 Conclusions

Considering the hardware characteristics of high-performance SIMD-DSP processors, this paper proposed an optimisation algorithm of the Cholesky decomposition. According to the structural characteristics of the result matrix of Cholesky decomposition, the LT matrix is generated, and the column update in the calculation process of generating the lower triangular matrix is replaced by the row update in the calculation process of generating the upper triangular matrix, which avoids the discontinuous access to the memory. While updating the matrix, the vector shuffling unit is used to realise the effect of broadcasting a single value as a vector. In this algorithm, the optimisation methods such as loop unrolling and software pipelining can also be used to develop better instruction-level parallelism. The experimental results on the FT-M7002 processor show that the optimised algorithm has a speedup ratio of 3.8~5.64 compared to the corresponding serial algorithm and a speedup ratio of 1.39~2.14 compared to the corresponding TI library functions. These results show that this paper's Cholesky decomposition optimisation algorithm is effective on SIMD-DSP processors. Though the optimisation algorithm of Cholesky decomposition in this paper is for the single-core case of vector DSP, it is helpful further to research the corresponding optimisation algorithm for multi-core situations.

Acknowledgements

This work was supported by Research Projects of Hunan Provincial Department of Education (No. 20B242 and No. 19A169) and Hunan Provincial Natural Science Foundation (No. 2017JJ3087).

References

Adriaansen, M., Wijtvliet, M., Jordans, R. et al. (2016) 'Code generation for reconfigurable explicit datapath architectures with LLVM', *IEEE Digital System Design*, pp.30–37, DOI: 10.1109/DSD.2016.88, https://ieeexplore.ieee.org/document/7723532.

Chen, Y., Wang, M.Y., Chai, X.N. et al. (2021) 'Optimization of Gaussian filtering algorithm on FT-M7002', *Computer Engineering*, Vol. 43, No. 5, pp.799–806.

Cupertino, L.F., Pacheco, M., Farias, R. et al. (2010) 'LU decomposition on GPUs: the impact of memory access', *IEEE International Symposium on Computer Architecture & High Performance Computing Workshops*, pp.19–24.

Dorris, J., YarKhan, A., Kurzak, J. et al. (2018) 'Task based Cholesky decomposition on Xeon Phi architectures using OpenMP', *International Journal of Computational Science and Engineering*, Vol. 17, No. 3, pp.310–323.

Gao, J.B. and Li, B. (2015) 'A Cholesky preconditioned conjugate gradient algorithm on GPU for the 3D parabolic equation', *International Journal of Computational Science and Engineering*, Vol. 11, No. 4, pp.339–348.

Golub, G. and Loan, A. (1986) 'Matrix computations', *Mathematical Gazette*, Vol. 83, No. 498, pp.556–557.

Haidar, A., Abdelfatah, A., Tomov, S. et al. (2017) 'High-performance Cholesky decomposition for GPU-only execution', *Proceedings of the General Purpose GPU*, pp.42–52.

Henrichs, P.M., Whitlock, L.R., Sochor, A.R. et al. (2009) *Cortex-A9 Technical Reference Manual*.

Herholz, P. and Sorkine-Hornung, O. (2020) 'Sparse Cholesky updates for interactive mesh parameterization', *ACM Trans. Graph*, Vol. 39, No. 6, pp.67–77.

Jin, Z.J. (2019) 'SAR target recognition based on Cholesky decomposition weighted kernel extreme learning machine', *Proceedings of the 2019 3rd International Conference on Advances in Image Processing*, Association for Computing Machinery, pp.40–44.

Lee, Y., Avizienis, R., Bishara, A. et al. (2016) 'The Maven vector-thread architecture', *2011 IEEE Hot Chips 23 Symposium (HCS)*, p.1.

Lemaitre, F., Couturier, B. and Lacassagne, L. (2018) 'Small SIMD matrices for CERN high throughput computing', *Proceedings of the 2018 4th Workshop on Programming Models for SIMD/Vector Processing*, pp.1–8.

Lin, Y., Lee, H., Who, M. et al. (2007) 'SODA: a high-performance DSP architecture for software-defined radio', *IEEE Micro*, Vol. 27, No. 1, pp.114–123.

Liu, S.Y., Lin, J.Y., Wu, Y.X. et al. (2016) 'Cholesky decomposition and parallel structure design based on matrix triangularization decomposition'. *Journal of Tsinghua University*, Vol. 56, No. 9, pp.963–968.

Nino-Ruiz, E.D., Sandu, A. and Deng, X.W. (2015) 'A parallel ensemble Kalman filter implementation based on modified Cholesky decomposition', *Proceedings of the 6th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, Association for Computing Machinery, pp.1–8.

Robelly, J.P., Cichon, G., Ahlendorf, H. et al. (2008) 'A HW/SW design methodology for embedded SIMD vector signal processors', *International Journal of Embedded Systems*, Vol. 3, No. 3, pp.160–169.

Rowen, C., Dan, N., Ravindran, R. et al. (2011) 'The world's fastest DSP core: breaking the 100 GMAC/s barrier', *2011 IEEE Hot Chips 23 Symposium (HCS)*, pp.1–25.

Shen, Y. and Dai, Y.X. (2019) 'Parallel Cholesky decomposition and its application based on GPU', *Computer Engineering*, Vol. 45, No. 2, pp.284–289.

Tang, B., Tang, M., Xia, Y. and Hsieh, M.Y. (2021) 'Composition pattern-aware web service recommendation based on depth factorisation machine', *Connection Science*, Vol. 33, No. 4, pp.870–890.

Wang, J., Sohl, J., Kraigher, O. et al. (2011) 'ePUMA: embedded parallel DSP processor architecture with unique memory access', *IEEE Information Communications and Signal Processing*, pp.1–5, https://ieeexplore.ieee.org/document/617351610.1109/ICICS.2011.6173516.

Wang, Y., Li, C., Liu, C. et al. (2021) 'Advancing DSP into HPC, AI, and beyond: challenges, mechanisms, and future directions', *CCF Transactions on High Performance Computing*, pp.114–125, https://doi.org/10.1007/s42514-020-00057-2.

Woh, M., Seo, S., Mahlke, S. et al. (2009) 'AnySP: anytime anywhere anyway signal processing', *ACM SIGARCH Computer Architecture News*, pp.128–139.

Xu, J., Ma, N., Ke, J., Yang, E.J. and Feng, S. (2019) 'A fast video haze removal algorithm via mixed transmissivity optimisation', *International Journal of Embedded Systems*, Vol. 11, No. 1, pp.84–93.

Yang, X., Tan, Z. and Luo, Z. (2021) 'Deep learning in mobile computing: architecture, applications, and future challenges', *Mobile Information Systems*, pp.1–3, https://doi.org/10.1155/2021/9874724.

Yu, A. (1996) 'The future of microprocessor', *IEEE Micro*, Vol. 54, No. 5, pp.67–77.

Zhou, P., Zhou, W.C. and Wang, K.K. (2014) 'Research on the performance of TMS320C6678 multicore DSP parallel memory access', *Microcomputer & its Applications*, Vol. 33, No. 13, pp.20–24.