Data Compression and Vectorization of Matrix Multiplication on HXDSP

Liu Yufu[1,*], *Lang* Wenhui[1], and *Jia* Guangshuai[2]

[1]Hefei University of Technology, School of Computer and Information, 230009, Hefei, China
[2]NO.38th Research Institute of China Electronic Technology Group Corporation, 230088, Hefei, China

Abstract. In order to solve the problem of low efficiency of hardware resources and low data processing ability of vector processors, this paper uses data compression and vectorization method to realize matrix multiplication based on the HXDSP platform with the DCT algorithm in HEVC. It can make full use of the hardware resources of DSP to achieve the optimal optimization. The experimental results show that this method can achieve 32GMACS which is the peak-point multiply-accumulate capability of HXDSP. It can achieve to 2Gpixel/s for the data processing capability, which meets the performance requirements of HEVC coding standard and provides a reference for hardware implementation of HEVC.

1 INTRODUCTION

In the field of digital signal processing, matrix multiplication is an important operation. In order to achieve highest computing performance on different hardware platforms, matrix multiplication requires a combination of the hardware platform architecture and its hardware resources, etc. At present, the method of matrix multiplication on the hardware platform can be summarized as follows: universal matrix multiplication[1-3], decomposition of matrix[4-10] and vectorization of matrix multiplication[11~15]. However, the performance of matrix multiplication on vector processors often can not reach its peak, which severely restricted computational performance of vector processors. How to effectively utilize the hardware resources of the vector processor has been a research focus in recent years.

In order to improve the matrix computing ability of DSP, the 38th Institute of China Electronics Technology Group has designed a vector processor-HXDSP1041 by adjusting the architecture of BWDSP100. Combining HXDSP1041 features of software and hardware, this paper adopts data compression and vectorization of matrix multiplication method which can make fully use the computing resources and data storage resources in HXDSP1041 and improve the number of data processing and data processing speed. According to the experimental results of the latest video coding standard HEVC on DSP, this paper verifies the effectiveness of proposed method on the HXDSP1041 with the implementation of DCT algorithm in HEVC.

2 HXDSP AND ITS FEATURES

HXDSP1041 is a high-performance 32-bit vector processor developed on the basis of BWDSP100 by the 38th Research Institute of China Electronics Technology Group Corporation. Its working frequency is 800MHz~1GHz. HXDSP1041 structure diagram shown in Fig1. It is integrated a new generation of core eC104+ which is the enhancement and improvement of eC104 in BWDSP100. At the frequency of 1GHz, the peak fixed-point computing performance is 84GOPS, and the peak floating-point computing performance is 84GFLOPS, and the peak fixed-point computing performance is 32GMACS. It mainly used in video image processing, radar signal processing, communications and electronic warfare and other fields.

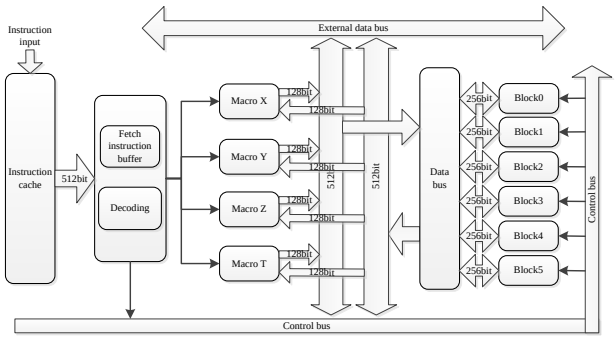


Fig1. HXDSP1041 structure diagram

The components that implement vector operations are composed of four internal vector processing units in

* Corresponding author: hfut_lyf@163.com

HXDSP1041. Each vector processing unit contains 8 multipliers (MUL), 8 adders (ALU), 4 shifters (SHF) and 1 super computing unit (SPU), as well as some common data registers and special data registers.

Each vector processing unit can calculate 21 times at the same time. So it can reach to 84 computations simultaneously with 4 vector processing units. When working with one vector unit, it can process 32 elements at a time. So we can make the best use of all of the resources in HXDSP1041 by arranging the data in integral multiple of 32. In addition, combining vector processing instruction parallelism, loop unrolling, and software pipelining etc. can further enhance its data processing capabilities.

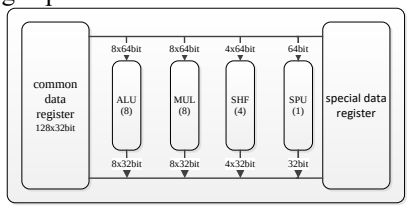


Fig2. HXDSP1041 vector processing unit

2.1 Instructions parallelism

Each basic computing unit of HXDSP1041 use SIMD mode which is an instruction can control multiple computing components to perform the same operation simultaneously. And the basic computing unit use MIMD. Each computing unit is controlled by a separate instruction to complete the computation independently. In HXDSP1041, each instruction can control up to 4 basic operation units. And up to 16 instructions can be executed in parallel in the same instruction cycle. Take full advantage of read and write bus and internal computing resources in DSP can greatly improve the efficiency of code, while reducing the amount of code.

2.2 Loop Unrolling

The more time-consuming parts of the code are generally concentrated in large loops, so the core of code optimization is optimization of the loop. Firstly, we must minimize the code inside the loop by moving the unrelated statements to the outside of the loop when writing the code. In addition, the most commonly used method of loop optimization is loop unrolling.

2.3 Software pipelining

Similar to the loop unrolling method, software pipelining also optimizes the code through the reorganization loop. However, instead of simply expanding the loop, the software pipelining intends to interleave the instructions of different recursive entities so that instructions between different recursive entities can be executed in parallel.

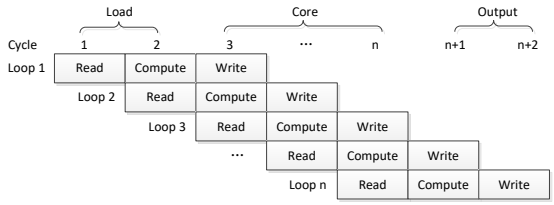


Fig3. Circulation body software flow diagram

In software pipelining, when some processing units run one loop, other processing units run another loop at the same time, we need to reduce the correlation between registers and operands in the loop body. Software pipelining can achieve both the full using of DSP hardware resources, but also reducing the cycle by reorganizing of the instructions of different recursive body.

2.4 Memory accessing of modulo 16

The memory accessing of modulo 16 is a scheme of memory accessing designed by HXDSP for matrix. The address is generated according to modulo 16 rules and the data memory is read or written by four words.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Fig4. Memory accessing of modulo 16

For example, for the instruction: {xRa + 3: ayRb + 3: bzRc + 3: ctRd + 3: d} = Dm [Un + = Um, Uk], the following four sets of addresses are firstly generated:
Un,Un+1,Un+2,Un+3;
Un+4*Uk,Un+4*Uk+1,Un+4*Uk+2,Un+4*Uk+3;
Un+8*Uk,Un+8*Uk+1,Un+8*Uk+2,Un+8*Uk+3;
Un+12*Uk,Un+12*Uk+1,Un+12*Uk+2,Un+12*Uk+3;

According to the address formed by this algorithm as the instruction of memory accessing of modulo 16, the data of the first group address are respectively sent to xRa, xRa + 1, xRa + 2 and xRa + 3. The second group address is respectively sent to xRb, xRb + 1, xRb + 2 and xRb + 3. The third group of addresses is respectively sent to xRc, xRc + 1, xRc + 2 and xRc + 3. The fourth group addresses is respectively sent to xRd, xRd + 1, xRd + 2, xRd + 3. And the Un is updated to Un = Un+Um after the instruction finishes.

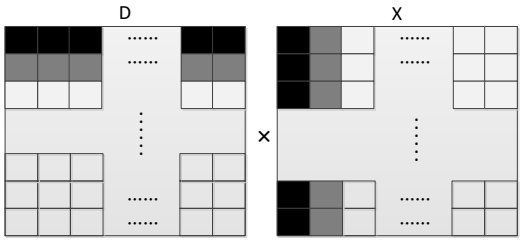
3 MATRIX MULTIPLICATION

Traditional matrix multiplication is calculated by means of multiplication of rows and columns and then accumulating. It can be described as:

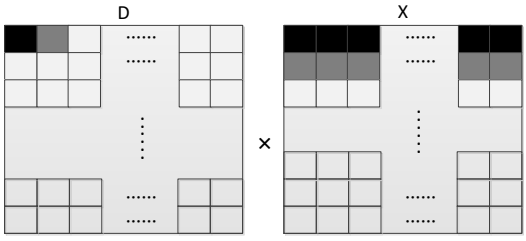
$$C_{ij} = \sum_{n=0}^{m-1} D_{in} \bullet X_{nj} \quad (1)$$

The algorithm of traditional matrix multiplication is relatively simple. That is the i-th row in the D matrix and the j-th column in the X matrix are correspondingly multiplied and accumulated. This algorithm of using advanced language is relatively easy to implement. However, if this algorithm of using assembly language is used on the DSP, it is more troublesome to take data by columns for the X matrix. Moreover, after the multiplication is completed, the data on different

processing units need to be accumulated, which limits the data processing capability of the DSP.



（a）The traditional matrix multiplication



（b）The vector matrix multiplication

Fig5. The comparison between the traditional matrix multiplication and the vector matrix multiplication

According to the structure of vector processor HXDSP1041, matrix multiplication use vector method instead of traditional method. The data of m-th element of the i-th row of the D matrix multiply and accumulate the m-th row elements of the X matrix. And repeat n times as above to complete the calculation of matrix multiplication. Fig5 shows the comparison between the traditional matrix multiplication and the vector matrix multiplication. The vector matrix multiplication can be described as:

$$\begin{cases} \overrightarrow{c_0} = d_{00}\bullet\overrightarrow{X_0} + d_{01}\bullet\overrightarrow{X_1} + \ldots + d_{0m-1}\bullet\overrightarrow{X_{m-1}} \\ \overrightarrow{c_1} = d_{10}\bullet X_0 + d_{11}\bullet X_1 + \ldots + d_{1m-1}\bullet X_{m-1} \\ \quad\vdots \\ \overrightarrow{c_{n-1}} = d_{n-10}\bullet\overrightarrow{X_0} + d_{n-11}\bullet\overrightarrow{X_1} + \ldots + d_{n-1m-1}\bullet\overrightarrow{X_{m-1}} \end{cases} \quad (2)$$

In Equation (2), each element in the matrix D will compute a multiplication with the corresponding row vector of the X matrix. If the matrix is too large, the number of multiplications to be computed will be greater. In order to reduce the using of multipliers, the data elements in matrix D are combined, that is, data compression. Data compression may be performed in two data compression forms, four data compression forms and so on, and the matrix X need to be compressed accordingly. The matrix multiplication of two data compression forms may be specifically described as follows:

$$\begin{cases} \overrightarrow{c_0} = \begin{bmatrix} d_{00} & d_{01} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_0} \\ \overrightarrow{X_1} \end{bmatrix} + \begin{bmatrix} d_{02} & d_{03} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_2} \\ \overrightarrow{X_3} \end{bmatrix} + \ldots + \begin{bmatrix} d_{0m-2} & d_{0m-1} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_{m-2}} \\ \overrightarrow{X_{m-1}} \end{bmatrix} \\ \overrightarrow{c_1} = \begin{bmatrix} d_{10} & d_{11} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_0} \\ \overrightarrow{X_1} \end{bmatrix} + \begin{bmatrix} d_{12} & d_{13} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_2} \\ \overrightarrow{X_3} \end{bmatrix} + \ldots + \begin{bmatrix} d_{1m-2} & d_{1m-1} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_{m-2}} \\ \overrightarrow{X_{m-1}} \end{bmatrix} \\ \quad\vdots \\ \overrightarrow{c_{n-1}} = \begin{bmatrix} d_{n-10} & d_{n-11} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_0} \\ \overrightarrow{X_1} \end{bmatrix} + \begin{bmatrix} d_{n-12} & d_{n-13} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_2} \\ \overrightarrow{X_3} \end{bmatrix} + \ldots + \begin{bmatrix} d_{n-1m-2} & d_{n-1m-1} \end{bmatrix}\bullet\begin{bmatrix} \overrightarrow{X_{m-2}} \\ \overrightarrow{X_{m-1}} \end{bmatrix} \end{cases} \quad (3)$$

With the improved method of (3), on the same vector processor, the speed of processing matrix multiplication can be doubled compared with (2) due to the reduction of data processing form. In addition, in the case of register without overflow, more data can be compressed according to the characteristics vector processor. It can greatly improve the data processing speed on vector processor.

4.MATRIX MULTIPLICATION OF DATA COMPRESSION AND VECTORIZATION

4.1 Computational complexity analysis of DCT in HEVC

Discrete Cosine Transform (DCT) is one of the most important means of removing the image redundancy by the latest video coding standard HEVC. It is also the best embodiment of the matrix operation. It is very important that the appropriate DCT algorithm is designed on the vector processor to realize the HEVC real-time. Because the DCT is one of the most time-consuming modules in HEVC, designing the DCT algorithm suitable can effectively reduce the encoding time on the HXDSP1041.

In the HEVC standard, the butterfly algorithm is adopted according to the characteristics of the DCT transform matrix, and 4 different size butterfly algorithms of 4×4, 8×8, 16×16 and 32×32 are used. The butterfly algorithm of 32×32 show in Fig6.

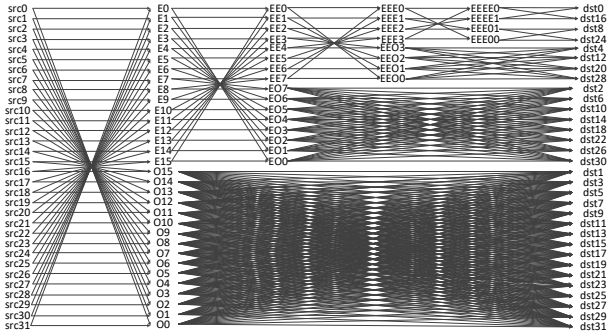


Fig6. 32×32 Butterfly algorithm

For the 32×32 matrix of DCT transform, using the traditional matrix multiplication algorithm requires 32×32×32=32768 multipliers and 31×32×32=31744 adders. However, using a butterfly algorithms requires only (16×16+8×8+4×4+4×2)×32 =11008 multipliers and (16×15+8×7+4×3+4×1+4×2+8×2+16×2)×32=11776 adders. Therefore, butterfly algorithm can save a lot of hardware resources, thereby increasing the speed of data processing.

Although the butterfly algorithm can save more hardware resources, but whether it is suitable for processing on vector processors, complexity analysis is also required. Butterfly algorithm uses the form of hierarchical calculation, the results of each level of operation has half data as the next level of input data. So it is not conducive to parallel computing because of the dependency between data and data in different level.In addition, as the matrix increases, the number of general data registers will be increased in each level increases. Usually, in order to solve the limitation of the number of general data registers on vector processors, batch calculation is usually performed. But it generates extra steps and time, and it will reduce processing speed.

However, there is no data dependency on vector matrix multiplication algorithm. Therefore, data compression and vectorization are very convenient for vector processor, because it can reduce the number of data in form.

4.2 Realization of Data Compression and Vector Matrix Multiplication

In order to ensure data correctly after compression, we neeed analysis data overflow problem based on the type of data. In HEVC, the range of pixel residuals is -255~255, and the coefficients in the transform matrix range from -90 to 90. Therefore, pixel residuals and the coefficients in transform matrix can all be stored in 16-bit data, so we can divide a 32-bit general register into two 16-bit registers. And we can divide one 80-bit MACC into two 40-bit MACC to store they results.

Table 1. Theoretical cycles of matrix multiplication (32 × 32)

Operation	Components	Numbers	Cycles
Read	2048	32	32
MUL	32768	32	512
Write	1024	32	16
Theoretical			560

According to the hardware resources of HXDSP1041, we can get the theoretical cycle of method proposed which shown in Table 1. Because the vector processing unit in HXDSP1041 contains a set of multiply-accumulate special registers which can replace adders, matrix multiplication can be realized only using multipliers. This is an advantage of vector processors.

In this paper, the step of proposed algorithm of DCT is as follows:

(1)Data compression. First, two data continuously which are 16-bit of each row in the DCT transform matrix are compressed one element which is 32-bit. And two data continuously of each column in the matrix X are compressed respectively.

(2)Read the data. All the data in the first row of the DCT transform matrix D and all the data from first row to m-th row of the matrix X are read from the memory (m is a positive integer which is less than or equal to the number of the matrix rows). Then, we extend m data of first row in the D matrix into one vector of the same value.

(3)Calculate and read new data. Multiply and accumulate from the first to m-th row data of the X matrix. In addition, In order to take fewer cycles, the next required data in the matrix X is read while the computation is in progress.

(4)Repeat step (3) until all the data in matrix X has been multiplied, then write the first row of result back to the memory.

(5)Similar to the above steps (2) to (4), we can get the final result of matrix multiplication.

5.EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

In order to verify the effectiveness of the our proposed method, we achieve 4×4 to 32×32 DCT transform on HXDSP1041 by different way and count of the number of execution cycles for each method, the results shown in Table 2. As can be seen from the table, the cycles of DCT which implemented by assembly language is significantly reduced compared to the cycles implemented by C language. For the DCT of 4×4 and 8×8, the cycles of butterfly algorithm are slightly less than the cycles of proposed method. However, for the DCT of 16 × 16 and 32 × 32, the disadvantage of the butterfly algorithm is exposed. And the cycles of proposed method are much less than the butterfly algorithm. Additionally, we can find that the cycles of the proposed method in 32×32 size is close to the theoretical analysis in Table 1.

Table 2. The cycles of DCT by different ways to achieve

Size	Buttefrfly (C)	Buttefrfly (ASM)	Buttefrfly-Data compress (ASM)	Proposed (ASM)
4×4	817	50	25	36
8×8	4146	208	95	97
16×16	44400	1063	583	280
32×32	292933	2627	1562	895

According to the experimental results, it is found that proposed method is most effective for the blocks with larger sizes for the DCT implementation of HEVC. Currently, in the live streaming video, because of its static background, the block processing with large size is particularly suitable. In this paper, combined with the hardware resources of HXDSP1041, if only consider the calculation part, the cycles of the calculation part is only 512 cycles in the 32×32 size matrix. Therefore, it can achieve 32GMACS which is the peak-point multiply-accumulate capability of HXDSP and it can achieve to 2Gpixel/s for the data processing capability.

6.CONCLUSION

In this paper, the method of matrix multiplication of data compression and vectorization is used to solve the problem of low hardware resource utilization and low data processing ability of matrix multiplication on vector processors. The implementation of DCT algorithm in HEVC shows that it can make full use of the calculation resources and data storage resources of HXDSP. It can achieve to the peak capability of multiply-accumulate for fixed-point data in HXDSP. That meets the performance requirements of HEVC coding standard and provides a reference for hardware implementation of HEVC. In the future, we will verify the universality of proposed method in a larger dimension.

ACKNOWLEDGMENTS

This thesis is supported by the 38th Institute of China Electronics Technology Group (W2017JSKF0213).

REFERENCES

1. Nath R, Tomov S, and Dongarra J, et al.. An improved MAGMA GEMM for fermi graphics processing units. International Journal of High Performance Computing Applications, 2010, 24(4): 511-515.
2. Tan G, Li L, and Triechle S, et al.. Fast implementation of DGEMM on Fermi GPU[C]. Proceedings of 2011 International Conference for High Performance Computing,Networking, Storage and Analysis. ACM, 2011: 35.
3. Liu, Gang, et al. Optimization of DGEMM Function for Loongson3B1500 Architecture. Journal of Chinese Computer Systems. 35.7(2014):1523-1527.
4. Jaiswal, Manish Kumar, and N. Chandrachoodan. FPGA-Based High-Performance and Scalable Block LU Decomposition Architecture. IEEE Transactions on Computers. 61.1(2011):60-72.
5. Michailidis P D, and Margaritis K G. Implementing parallel LU factorization with pipelining on a multicore using OpenMP[C]. Computational Science and Engi-neering(CSE),2010 IEEE 13th International Conference on. IEEE, 2010: 253-260.
6. Venetis I E, and Gao G R. Mapping the LU decomposition on a many-core architecture: challenges and solutions[C]. Proceedings of the 6th ACM Conference on Computing Frontiers. ACM, 2009: 71-80.
7. TANG Yun. Study and implementation on distributed large scale matrix computation algorithms with Spark[D]. Nanjing University,2016.
8. YANG Fei, and MA Yuchun, and HOU Jin, et al.. Research on acceleratin of matrix multiplication based on parallel scheduling on MPSoC. Computer Science, 2017, 44(8):36-41.
9. LONG Zhuoqun, WANG Xiaoyu, and WANG Changming. Epiphany-OpenCL large matrix multiplication parallel computation method based on DCT predictive coding. Automation & Instrucmentation, 32.7(2017):16-21.
10. SHEN Junzhong, Xiao Tao, and QIAO Yuran, et al.. A matrix multiplicationaccelerator design for optimization blocking strategy. Computer Engineering & Science, 38.9(2016):1748-1754.
11. SHEN Junzhong, Xiao Tao, and QIAO Yuran, et al.. A matrix multiplicationaccelerator design for optimization blocking strategy. Computer Engineering & Science, 2016, 38(9):1748-1754.
12. WEI Shuai. Research on vectorization algorithm and reorganization technology for SIMD[D]. PLA Information Engineering University, 2012.
13. ZHANG Kai. High efficient matrix operations on vector-SIMD DSPs[D].National University of Defence Technology,2013.
14. Zhu, H., et al. Optimization of matrix multiplication based on a multi-core architecture extended with vector units. Journal of University of Science & Technology of China 41.2(2011):173-182.
15. Wang Jie. The implementation of a high performance vector processor [D].TianJin University,2016.
16. LIU Zhong, TIAN Xi. Vectorization of matrix multiplication for multi-core vector processors*, 2017, Vol.40, Online Publishing No. 94.