Acta Technica *62* (2017), No. 6B, 131–142

Local parallel mutation irregular code generation of data flow driven

Guofang Li[1], Ning Lu[2]

Abstract. In order to improve parallel computational efficiency and accuracy of large matrix multiplication, a kind of Epiphany-OpenCL parallel computational mode of large matrix multiplication has been proposed based on DCT predictive coding, which has been proposed in this paper. DCT transformed value of two-dimensional data and its inverse transformation has been used to realize the dimensionality reduction of matrix data, at the same time, OpenCL parallel transform coding process has been designed based on Epiphany to realize parallel processing of matrix multiplication. Experimental results have verified effectiveness of the proposed mode. In realization of the said algorithm, Zynq series chips of Xilinx have been used for verification.

Key words. Predictive coding, Epiphany architecture, Parallel computation of matrix, Local code, Inordinance.

1. Introduction

Epiphany connects different computational kernels with two-dimensional mesh network structure so that data [1-3] is exchanged between different kernels through two-dimensional mesh network. Two types of products including 16 kernels and 64 kernels have been developed in this platform, which can be expanded [4] through interconnection between boards. Performance of an Epiphany IV processor is 50GFLOPS/W, which is the most efficient one in parallel processors based on multi-purpose kernel. This architecture can be expanded to 4096 kernels (exascale computing level has been reached). Energy efficiency of Epiphany architecture has been certified as the guidance and future direction [5] of this kind of platform. Future processors will have hundreds of on-chip kernels, which is exactly consistent with Epiphany architecture characteristics. However, unlike the general-purpose processor, this processor has not been provided with cache since it is limited to the area and resources on

[1]Department of public infrastructure , Guangzhou Health Science College, Guangzhou, 510450 China

[2]Computer Engineering technical College, Guangdong Polytechnic of Science and Technology, Zhuhai, 510450 China

chip and the programming model is different from [6] the general programming way, too. Although Epiphany processor has higher energy efficiency and computational capacity, programmers still need to optimize the program with pertinence for giving full play to computational capacity. This paper has selected the representative matrix multiplication algorithm to research programming optimization of Epiphany processor. Matrix multiplication is widely used in the field of scientific computation, which has higher requirements for floating-point performance and is able to evaluate computational efficiency [7] of processor well. This paper has optimized data portioning, data communication between kernels and other relevant aspects aiming at Epiphany architecture characteristics. Epiphany supports a variety of programming models such as standard C, OpenCL, OpenMP and MPI. Aiming at protogenetic C language programming and relying on eSDK provided byAdapteva, this programming mode is able to give good play to Epiphany performance and support various programming. Although this programming mode is limited in used of Epiphany hardware, OpenCL, OpenMP and MPI program can be run between different computing platforms, which is very important[8∼9] in the field of heterogeneous computation [8∼9].

2. DCT predictive coding

2.1. Predictive coding technology

Predictive coding algorithm realizes elimination of adjacent data correlation in the space domain and time domain to obtain excellent data compression effect [10∼11] in the way of prediction to data without coding, as shown in Fig. 1. Firstly, $x(n)$ is set as the current input data to realize the prediction to $x(n)$ value through adjacent coding data and the predictive value is $p(n)$; secondly, entropy coding quantization shall be implemented to deviation $e(n)$ between predicted value $p(n)$ and true value $x(n)$; finally reconstruction value $x(n)$ can be obtained based on quantization of residual $e'(n)$ and predicted value $p(n)$, which can be expressed as $x'(n)$ and act as the hypothetical value of data without coding.

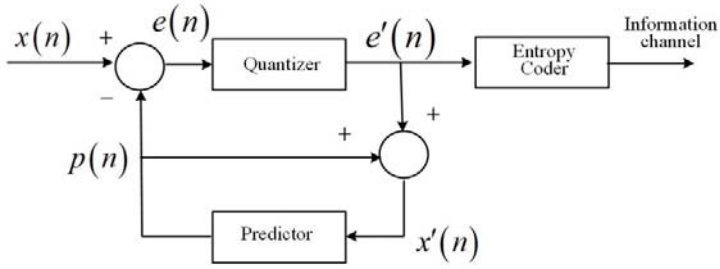

Fig. 1. Basic process of predictive coding

There are mainly two kinds of predictive coding: (1) prediction in data, mainly for redundancy elimination of spatial domain; (2) prediction between data, mainly

for redundancy elimination of time domain. However, data compression can be realized only based on prediction technology if limit elimination of time redundancy is used in data.

2.2. Discrete cosine coding

According to Discrete Fourier Transform (DFT), it can be known that any data signal can be expressed as superposition of multiple sine and cosine data with different frequencies and amplitudes. If input data is discrete and consistent with cosine form, then decomposition of data signal is discrete cosine process (Discrete Cosine Transform, DCT). In the field of mathematical research, there are many different types of DCT, but all of them belong to the field of real even DFT. Research contents in this paper mainly involves relevant coding problems and DCT used hereof is the DCT transform process of II type for simplifying expression of the problem. Computational formula [12∼13] one-dimensional DCT transform process used in this paper is shown as follows:

$$X(k) = \sqrt{\frac{2}{N}}\varepsilon_k \sum_{n=0}^{N-1} x(n) \cos\left[\frac{2N - 1k(2n+1)\pi}{2N}\right]. \tag{1}$$

In the formula (1), $k = 0, 1, \cdots, N-1$; when $k = 0$ or $k = N$, value of ε_k is $1/\sqrt{2}$, otherwise its value is 1; N is data quantity in data sequence and value range of k and n is $[0, N-1]$; $X(n)$ is as a one-dimensional data in data sequence. According to above formula, it can be seen that if $k = 0$, then it will be expressed as the direct current signal component (Direct Current, DC) and in this case, $X(0)$ is in direct proportion to signal $X(n)$ mean; if $k > 0$, then it will be expressed as the alternating current signal component (Alternate Current, AC) and in this case, the overall change trend of AC will speed up with growth of k in the sensitiveness of $X(k)$ on frequency of $X(n)$, which will lead higher growth trend of frequency value. The detailed computational demonstration is shown as:

$$\begin{cases} x(m): \ 56, 38, -25, -15, 11, -67, 14, 39\,, \\ y(n): \ 51, 53, 50, 49, 48, 52, 47, 50\,. \end{cases}$$

Based on $y(n)$ and $x(m)$ with different characteristics as mentioned above and use of DCT transform, DCT transform characteristics as presented shall be analyzed and researched, with specific process as shown as:

It is assumed that the one-dimensional sequence $x(n)$ consist of 8 sets of data, namely: 56, 38, -25, -67, -26, 14, 39. If $k = 0$, then value of $X(0)$ can be computed according to formula (2).

$$X(0) = \frac{1}{2\sqrt{2}}(56 + 38 - 25 + 11 - 67 - 26 + 14 + 39) \approx 15\,. \tag{2}$$

If $k = 1$, the following can be obtained according to formula (2):

$$\begin{aligned} X(1) =& \frac{1}{2}(56\cos\frac{\pi}{16} + 38\cos\frac{3\pi}{16} - 25\cos\frac{5\pi}{16} + 11\cos\frac{7\pi}{16} \\ &+ 67\cos\frac{7\pi}{16} + 26\cos\frac{5\pi}{16} - 14\cos\frac{3\pi}{16} - 39\cos\frac{\pi}{16}) \approx 23\,. \end{aligned} \tag{3}$$

All the data DCT values in the data sequence can be obtained further according to the above process. DCT transform result can be obtained with comparison to sequential data with large interval. It can be seen that data sequence with wider range of distribution has a weak correlation while data sequence with adjacent distribution has a strong correlation. Therefore, DCT transform value will cause the energy of the data sequence to be concentrated in the DC coefficient when $k = 0$. When $k > 0$, AC coefficient has small amplitude. This mode realizing signal concentration is able to facilitate data entropy encoding and quantization to realize the optimization of data signal. Two-dimensional expression of the transform process of matrix DCT is shown in formula (4).

$$X(k,1) = C(k)C(1)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)\cos\left[\frac{(2m+1)k\pi}{2N}\right]\cos\left[\frac{(2n+1)l\pi}{2N}\right]. \tag{4}$$

And then two-dimensional inverse transformation is shown in formula (5).

$$x(m,n) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} C(k)C(l)x(k,l)\cos\left[\frac{(2m+1)k\pi}{2N}\right]\cos\left[\frac{(2n+1)l\pi}{2N}\right]. \tag{5}$$

In the formula, when values of parameters k, l are 0, values of coefficients $C(k)$ and $C(l)$ are $\sqrt{1/N}$ and the values in other cases are selected as $\sqrt{2/N}$; value range of parameters m, n, k, l can be expressed as $[0, 1, 2, 3, \cdots\cdots, N-1]$.

3. OpenCL parallel transform coding process based on Epiphany

3.1. Programming model and architecture

Schematic diagram [14∼15] of Epiphany architecture system used in this paper is shown in Fig. 2. The architecture form used hereof is distributed and scalable memory and share computer system, which is composed of two-dimensional node array in the form of processor and in which a low-delay eMesh can be used to connect on-chip network data connection between processors. Every processor in this architecture can be deemed as instruction set CPU and the superscalar floating-point is able to process 64 bits memory access processes and two floating-point operation processes in every clock period. On every group of routers of eMesh, three channels can be opened for data communication: channel written off ship, channel written on ship and reading channel. It can support the data communication between the

kernels so that the communication delay is usually nanosecond.

In Epiphany architecture, the processor generally consists of two layers of memorizer: (1) off-chip memory is located within shared space of outer layer of processor, at size less than 1GB; (2) local memory is located within the inner layer of the processor, with computational nodes at size of 32KB. It shall be recognized that all the computational nodes in architecture can use network on two-dimensional chip to access the router of adjacent computational nodes within local memory space. Since size of local processing memory is only 32KB, which has higher restriction on data communication and task partition.

In order to make full use of processing capacity of computational node, off-chip memory is needed to be made full use to expand computational space and data size of off-chip memory access shall be reduced as far as possible at the same time. Since processing node of Epiphany architecture used is not provided with mainframe processing function, it can be used as coordinating processor in the process of data processing.

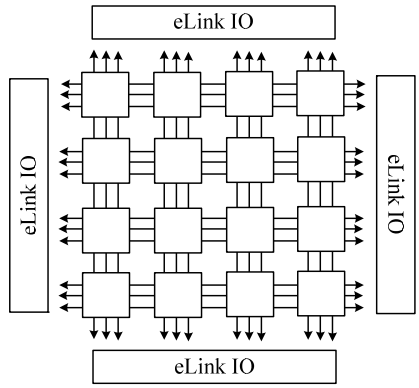

Fig. 2. Epiphany computational architecture

3.2. OpenCL parallel computation of matrix multiplication

It is assumed that n-dimensional vector $x = [x_1, x_2, \cdots, x_n]^T$ and n- order vector matrix is A and parallel computation is implemented to product of matrix A and vector x to acquire $y = [y_1, y_2, \cdots, y_n]^T$. OpenCL model shall be used for parallel multiplication computation of matrix herein. Partition shall be implemented to matrix based on one-dimensional partition and the full column (full row) of matrix group can be acquired in this way. Then a processor shall be used to be responsible for processing and memory of a group. The said block parted can be in isometric or continuous forms to represent two kinds of block partition forms including continuous band or cyclic band. In this research, continuous block partition has been implemented to matrix A in the way of column partition to obtain n data blocks, then the number of row of every data block can be deduced to be n/p. At the same time, every row block has been parted into p sub-blocks correspondingly, then

$[A_{k,0}A_{k,1}\cdots A_{k,p-1}]$ can be acquired after matrix partition implemented to A_k, the row block k of matrix$A$. Thus, expression of the above block $A_{k,j}$ is shown as follows:

$$A_{k,j}=\begin{bmatrix} A_{kn/p+1,jn/p+1} & A_{kn/p+1,jn/p+2} & \cdots & A_{kn/p+1,jn/p+n/p} \\ A_{kn/p+2,jn/p+1} & A_{kn/p+2,jn/p+2} & \cdots & A_{kn/p+2,jn/p+n/p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{kn/p+n/p,jn/p+1} & A_{kn/p+n/p,jn/p+2} & \cdots & A_{kn/p+n/p,jn/p+n/p} \end{bmatrix}. \quad (6)$$

In formula (6), continuous row partition has been implemented to matrix A based on the subscript while corresponding processor quantity has been used for column partition to subscript j of matrix A, of which the greatest characteristic is to be related to processor quantity p and the partitioning range is $[0,1,\cdots,p-1]$. Matrix A is the n-order matrix and its sub-block has n/p orders after partitioning. Similarly, n-order sub-block partition has been implemented to subsequent product vector x and the result vector y in the form of row partitioning. Then the results are shown as follows:

$$\begin{cases} x=[x_0,x_1,\cdots,x_{p-1}]^T, \\ y=[y_0,y_1,\cdots,y_{p-1}]^T, \end{cases} \quad (7)$$

Its corresponding sub-block k is expressed as:

$$\begin{cases} x_k=[x_{kn/p+1},x_{kn/p+2},\cdots,x_{kn/p+n/p}]^T \\ y_k=[y_{kn/p+1},y_{kn/p+2},\cdots,y_{kn/p+n/p}]^T \end{cases} \quad (8)$$

After sub-block partition based on the processor quantity, it can be deduced that specific computation of product y_k resulted from matrix partition on processor k can be expressed as:

$$y_k=A_kx=\sum_{j=0}^{p-1}(A_{kj}x_j)=\sum_{j=0}^{p-l}(A_{k,(k+1)\bmod p}X_{j(k+j)\bmod p}). \quad (9)$$

Computational results of subscript module $(k+j)\bmod x_k$ of matrix in moving processing of matrix column subscript j according to $[0,1,\cdots,p-1]$ are $[k,k+1,\cdots,k-1]$. In the processing, computation shall be implemented first to sub-block A_{k,kx_k} located at local computational processor which shall be followed by $A_{k,(k+1)x_{k+1}},\cdots$ in turn. In this way, circulatory movement can be realized to matrix block computation until circulatory computation of the whole block matrix ends, in which traversal implementation has been realized to all the sub-blocks of matrix x.

If row coordinate corresponding to sub-block to be processed in the matrix A is always k and keep constant, sub-block used in matrix A will use the current processor in running status for memory in the process of using the above formula. Subscript of sub-block vector corresponding to vector x can be computed as $(k+j)\bmod p$, of which value is related to change of j, namely, traversal processing is needed for sub-block vector on vector x in the computational process. In the process of

implementation, vector x can be moved upward to realize circular implementation in processing.

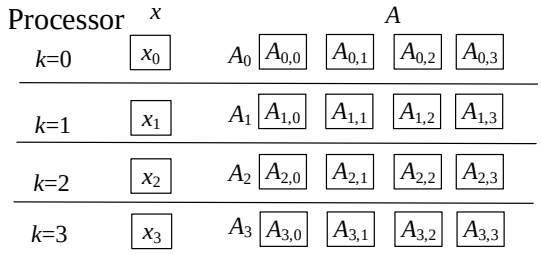

Fig. 3. Parallel product of matrix vector

In the schematic process of parallel production computation of matrix vector as shown in Fig.3, product operation shall be implemented to current memory sub-block x_k of vector x in processor and memory A_k of matrix A in the processor, for p times in total. Every product operation of sub-vector in vector x shall use round-robin mode toward the last step and accumulation of p times of processor product operations to acquire the result vector y of the final vector matrix product.

3.3. OpenCL data transmission of matrix multiplication algorithm

Main purpose of interface function API provided to parallel computation of matrix in eSDK is to acquire transmission capacity of external data shared and memorized on kernel and between kernels and use OpenCL supported by it to for hardware implementation to scalable algorithm of parallel multiplication of matrix proposed in Literature [10] so that characteristics of hardware Epiphany can be used fully.

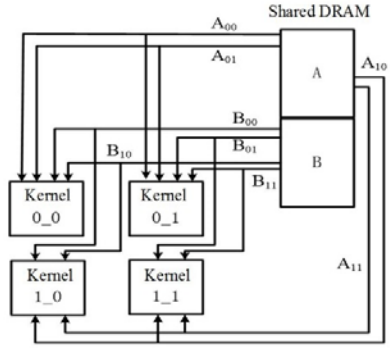

Fig. 4. OpenCL data transmission process of matrix multiplication

Epiphany architecture processor used in this research includes 16 computational nodes and uses the structural constitution as 4×4 which has been limited by on-chip memory condition. Upper limited of block of cokernel in matrix is 32×32 and then Epiphany processor can upload the matrix at size of 128×128 in shared external

memory in each process of ordinal data transmission. Where data between kernels is transferred on Epiphany architecture for three times, total times of performable matrix multiplication computation is 128×128.

4. Experiment and analysis

4.1. Experimental setup

Emphasis of this experiment is the hardware implementation of the targeted processor aiming at matrix multiplication algorithm realized on the basis of OpenCL as mentioned above and to obtain performance evaluation. Performance comparison parameters of ARM A9 processor and the targeted processor parameters are shown [16] in Table 1.

Table 1. Performance comparison parameter

Comparative parameter	Epiphany	ARM A9
Number of kernel (No.)	18	5
Clock frequency (MHz)	600	1200
Performance peak	19200	19200
Energy consumption (mW)	270	1350
Superficial area of chip (nm2)	2.05	4.6
MFLOPS(nm2)	9400	42000

Fig. 5. Computational chip based on epiphany

Epiphany processor used has frequency of 600MHz, performance peak of 19200, computational power consumption less than 270mW, computational ratio of energy computation more than 71000 and used computational chip as shown in Fig. 5. This kernel control chip is the Zynq series chip launched by Xilinx in April 2010.

4.2. Comparative evaluation of general programming model

Parallel matrix multiplication operation to be realized in this research on the basis of OpenCL is established under Epiphany framework, of which performance expression is shown with performance analysis data in Table 2.

Table 2. Algorithm performance analysis on epiphany

Matrix size	ANSICC	OpenMP	Improved MPI	MPI	OpenCL
64	0.15ms	21367.0ms	528ms	15637ms	653ms
128	0.26ms	35945.0 ms	898ms	24539 ms	1200ms
256	13.92ms	155865 ms	3952ms	125865 ms	4120ms

According to performance analysis and evaluation data on Epiphany as shown in Table 2, it can be seen that MPI programming mode is suitable for traditional multiplication operation of matrix without block, where computational time does not include times needed in initial operation to equipment and computing platform. Basic frequency of used processor is 667MHz. It can be seen from Table 2 that in the process of programming based on MPI programming mode, acquired computational performance is not ideal and the computational time is more than that of serial mode by above 2 time order of magnitudes. In order realize correspondence to ANSICC programming mode, block computation shall be used. Memory of every processor kernel on Epiphany architecture shall be defined in the form of private memory. Programming model needs to select data block within shared external memory space continuously to realize block computation to matrix multiplication operation. Compared with ANSICC programming process, block computation operation with use of OpenCL is much small than operations of ANSICC, MPI and other programming modes, which can reduce sharply expenditure in off-chip data access and memory. At the same time, DMAengine can be called to data transmission mode by using OpenCL to enhance efficiency of data transmission. Therefore, higher performance can be acquired by using OpenCL programming mode than those by using ANSICC, MPI and other programming modes.

4.3. Experimental analysis of large matrix multiplication

This experiment has selected test object of matrix multiplication at size 512×512, experimental mainframe of intel i7-6400HQ processor, experimental platform of Epiphany single-kernel chip, serial computing mode to optimize the parallel program on Epiphany chipset with 16 kernels. Comparison of computing time of desktop computer, single-kernel chip and 16-kernel chip is shown in Table 3 and the mean of 10 times of experiments has been taken for comparative data in this experiment to stabilize results in the computational process.

Table 3. Comparison of computing time of matrix multiplication

Evaluation index	Desktop computer	Single-kernel chip	16-kernel Epiphany chip
Total time	2451.3	532.8	354.9

Compared with the ordinary desktop mainframe, the biggest advantage of Epiphany parallel computational process is that it can obtain a lower balanced clock frequency in the system while computational efficiency can be increased by 5 times. In particular, computational efficiency with use of 16-kernel chip is higher, which

has reflected advantage in efficiency of Epiphany parallel computational process.

Time measuring results at different computational stages in Epiphany parallel computational process with use of 16-kernel chip are shown with comparative data in Table 4.

Table 4. Computational times of multi-kernel epiphany at different stages

Computational stage	Computational time	Percentage
Matrix multiplication of sub-block within kernel	18.3	11.9
Internal exchange of data	0.79	0.5
External DRAM read	134.6	87.6
Total time	153.7	100

According to computational time data of multi-kernel Epiphany at different stages as shown in Table 4, it can be seen: (1) total computational time does not equal to sum of times in various items and multi-kernel parallel computational performance is low; (2) the effective computational time used in matrix multiplication only accounts for about 10% of the total time and most of computational time is spent in reading and writing DRAM data.

5. Conclusion

In this paper, Epiphany-OpenCL parallel computation of large matrix multiplication based on DCT predictive coding has been realized according to the two kinds of programming models. Experimental results indicate that the proposed algorithm routine can acquire excellent performance while use of OpenCL routine can facilitate acquisition of better performance compared with uses of OpenMP, MPI and other routines. Moreover, this solution will be more popular with constant perfection in support of various programming algorithm models by Epiphany. In future, the popularization and application of this algorithm will be stressed.

References

[1] M. Ravishankar, J. Eisenlohr, L. N. Pouchet, et al.: *Code generation for parallel execution of a class of irregular loops on distributed memory systems*[C]// High PERFORMANCE Computing, Networking, Storage and Analysis. IEEE, (2012), 1–11.

[2] C. Hu, J. Li, J. Wang, et al.: *Communication Generation for Irregular Parallel Applications*[C]// International Symposium on Parallel Computing in Electrical Engineering, (2006). Par Elec. IEEE, 263–270.

[3] J. Lin, X. Tian, Ng J: *Mis-speculation-Driven Compiler Framework for Aggressive Loop Automatic Parallelization*[C]// IEEE, International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. IEEE Computer Society (2013), 1159–1168.

[4] M. Belaoucha, D. Barthou, A;Eliche, et al.: *FADAlib: An open source C++ library for fuzzy Array dataflow analysis*[J]. Procedia Computer Science, *1* (2010), No. 1, 2075–2084.

[5] L. A. J. Marzulo, T. A. O. Alves, F. M. G. França, et al.: *Couillard: Parallel Programming via Coarse-Grained Data-Flow Compilation*[J]. Parallel Computing, *40* (2011), No. 10, 661–680.

[6] D. B. Larkins: *Improving data locality for irregular partitioned global address space parallel programs*[C]// Southeast Regional Conference. ACM, (2012), 280–285.

[7] F. Li, A. Pop, A. Cohen: *Automatic Extraction of Coarse-Grained Data-Flow Threads from Imperative Programs*[J]. IEEE Micro, *32* (2012), No. 4, 190-31.

[8] S. Kim, H. Han: *Efficient SIMD code generation for irregular kernels*[J]. ACM SIGPLAN Notices, 47 (2012), No. 8, 55–64.

[9] D. Cho, S. Pasricha, I. Issenin, et al.: *Compiler driven data layout optimization for regular/irregular array access patterns*[J]. Acm Sigplan Notices, *43* (2008), No. 7: 41–50.

[10] V. Subotic, J. C. Sancho, J. Labarta, et al.: *Identifying Critical Code Sections in Dataflow Programming Models*[C]// Euromicro International Conference on Parallel, Distributed and Network-Based Processing. IEEE, (2013), 29–37.

[11] A. Smyk, M. Tudruj: *Hierarchical Optimization of the Parallel FDTD Computations Based on the Macro Data Flow Graph Paradigm*[C]// International Symposium on Parallel and Distributed Computing. IEEE, (2007), 10–10.

[12] D. Burger, S. W. Keckler, K. S. Mckinley, et al. *Scaling to the End of Silicon with EDGE Architectures*[J]. Computer, *37* (2004), No. 7, 44–55.

Received May 7, 2017