Transactions on Computational and Scientific Methods | Vol. 4, No. 11, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Advancing the Ecosystem of Domestic Processors: Development and Optimization of Function Libraries for Improved Performance

Quinlan Morrow
University of Colorado Denver, United States
quinlan23@ucden.edu

Abstract: The rapid advancement of the computer industry and the increasing need for information security have underscored the strategic importance of China's domestic processor ecosystem, especially in light of international trade restrictions. This paper explores the challenges faced by China's processor industry, including limited software ecosystem support, inadequate optimization for critical applications, and dependence on foreign technologies. By focusing on Kunpeng, Feiteng, and Loongson processors, this study highlights their potential and the urgent need for ecological development to enhance their competitiveness. Open-source software applications, such as CFL3D for fluid mechanics and GIMP for image processing, are analyzed for their role in advancing domestic processor ecosystems. Using tools like Turbo C and Authorware, a high-performance library of basic functions was developed, with implementations in C and assembly languages for DSP chips such as the FT-M7002. The research also demonstrates the successful deployment of optimized image and signal processing libraries on Huawei Kunpeng servers. These efforts mark a significant step towards addressing the "core shortage" problem and promoting the localization of China's processor industry.

Keywords: Chip; Algorithm Optimization; Database.

1. Introduction

1.1 Research Background

From the "516 Event" in 2019 to the "515 Event" in 2020, the United States has been pounding at the weakness of the underdevelopment of China's chip industry in an attempt to curb the rise of China. With the continuous development of the computer industry, the country has an urgent need for information security. However, if there is no localization and localization of CPU, this security will also be impossible. It has been 20 years since China started the processor research project in 2001. Although it has achieved good results, the domestic "core shortage" problem is still very serious in the face of the sanctions of the United States [1]. In order to solve the problem of "lack of core", the State Council issued the Several Policies for Encouraging the Development of Software Industry and Integrated Circuit Industry, which requires accelerating the industrialization and promotion and application of technologies with independent intellectual property rights [2]. In the face of huge pressure from all aspects, the domestic chip industry, whether it is free from the restrictions of foreign technology or the improvement of the ecological industry chain, is the key factor for its future success in the market. At present, there are six major processor manufacturers in China, which are Kunpeng, Feiteng, Haiguang, Longxin, Zhaoxin and Shenwei. Among the six processor manufacturers, Kunpeng, Feiteng and Loongson currently have great development potential: Kunpeng and Feiteng are based on ARM architecture level authorization, with high degree of autonomy, constantly

enriched ARM application ecology, broad market space and leading position in product performance; The LoongArch instruction set architecture independently developed by Loongson Zhongke has avoided being "stuck" in technology. Although it is inferior to Kunpeng and Feiteng in terms of ecological construction, with the improvement of the ecosystem, Loongson's market competitiveness cannot be underestimated.

All major manufacturers attach great importance to ecological issues. Domestic enterprises strive to achieve results in ecological construction through various ways, such as the "wisdom plan" being implemented by Kunpeng. Through the way of task package announcement, we invite university teachers and students, scientific research institutions, business partners and other developers to actively contribute wisdom, jointly build and improve Kunpeng's basic software capabilities, and realize Kunpeng's ecological co-construction and sharing [3]. CEC refers to the "Wintel" system and the "AA model" built by Intel and Microsoft to build the "PK system" (Feiteng Phytium+Kylin). At present, the system has been promoted to version 2.0, which plays a key role in software and hardware optimization and adaptation [4]. However, ecological problems can not be solved overnight. For example, the software ecology of Feiteng and Godson processors does not support mainstream software such as MySQL, Apache, Memcached, Hadoop, and there is also a huge demand for library functions. The huge gap in the ecological industry requires not only the promotion of the country from top to bottom and the vertical coordination of the whole industry, but also the resistance to the international ecological attack. The construction difficulty is far higher than expected.

Nowadays, more and more enterprises in the market have begun to focus on the construction of the domestic processor ecosystem, and have achieved many achievements, such as the optimization and implementation of the computer graphics display system of the domestic Feiteng 1500A processor by the University of National Defense Science and Technology, the C++intelligent source code conversion framework based on the domestic heterogeneous multi-core processor launched by the Qingdao Marine Science and Technology Pilot National Laboratory, and so on [5, 6]. But to meet the needs of the market, more development groups and better solutions are needed. The company's team aims to improve the basic ecological library in such fields as image processing, scientific computing, fluid mechanics, acoustic optical signal processing and other fields through more scientific and efficient methods for the three major domestic processors: Kunpeng, Feiteng and Longxin, and transplant and optimize open source software [7]. This project is conducive to accelerating the localization of the ecological reservoir and providing strong support for the development of the entire processor industry.

1.2 Problems and Challenges

Although China's processor product technology research and development has entered a high-speed development stage of simultaneous advancement of multi-technology routes, driven by major national science and technology projects and investment funds in the state-level integrated circuit industry, this does not mean that China's future processor development path will be smooth sailing, and Chinese enterprises still face many challenges and difficulties in the field of chips and processors[8].

Chen Ning, chairman and CEO of Shenzhen Yuntian Lifei Technology Co., Ltd., said in an interview, "The chip ecology is the key problem for intelligent robots to meet the opportunity." In fact, most domestic processors in the market today are more or less faced with the problem of being "stuck" due to inadequate ecological construction. For example, Kunpeng processor based on ARM architecture, although ARM architecture has achieved great success in mobile terminals such as mobile phones and tablets, it is still a new laggard in PC and server terminals, and it still needs time to accumulate in ecological construction. A large number of software manufacturers lack support for ARM architecture, which causes users to encounter various problems such as operating system, development environment and application software when deploying products and services on the server side, and cannot ensure the stable development of the company's business; Feiteng is the only chip manufacturer that has fully defined and implemented the architecture specification of the security processor platform. Although its ecological construction has made rapid progress in the past two years, its ecological construction is far from comparable with that of other international giants because of its late start, and it has no strength to establish a fully independent software and

hardware ecosystem in the field of general computing; There is also Godson processor, which has built an open ecosystemof the whole industrial chain from end to cloud, and has opened up hardware to facilitate participationin ecological construction.

However, its software ecosystem is weak, its application software and system software are not optimized enough, it does not support mobile ecosystem, its virtualization capability is insufficient, and it supports fewer cloud platform manufacturers [9,10]. It can be seen that it is urgent to speed up the ecological construction of domestic processors. Only by establishinga good ecological soil, can China's relevant enterprises develop better.

1.3 7002 Architecture Introduction

The FT-M7002 mainly includes one CPU core and two FT-MT2 DSP cores. The transmitted VLIW structure includes 5 outgoing scalar processing units SPU and 6 outgoing vector processing units VPU. The SPU includes instruction flow control, SPE (including 2 MAC and 1 IEU, which support fixed-point, single-precision and double-precision floating-point multiplication and addition) and SM (scalar data access)[11]. VPU consists of 16 isomorphic VPEs and mixing/protocol components. Each VPU is internally integrated with 3 MAC and 1 IEU to support fixed-point and floating-point calculations. DMA receives the transmission parameters configured by SPU to start access to specific storage resources, providing high-speed data transmission path for the kerne[12]l. AM is an on-chip array memory, which implements 16-channel SIMD-wide vector data access, and supports two vector storage instructions and DMA parallel access operations. Core level code refers to the code that has been imported into the chip through DMA and is executed based on the kernel operation unit[13].

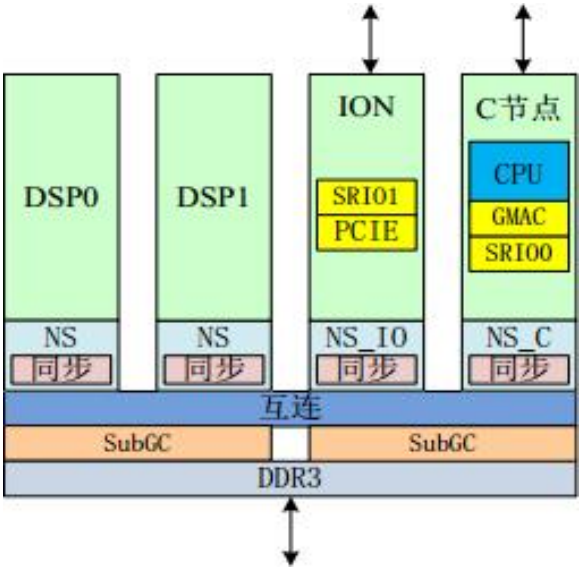


Figure 1. FT-M7002 architecture

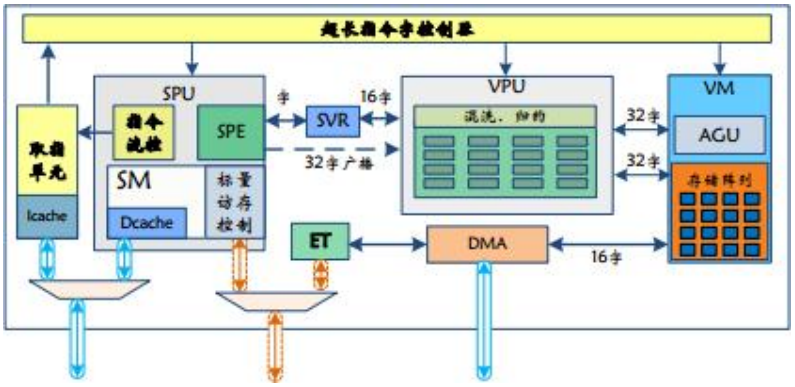


Figure 2. FT-MT2 Core Structure

2. Matrix Multiplication Algorithm Real Column

The matrix multiplication is calculated as C = A * B. The A matrix scale is M*K, the B matrix scale is K*N, and the C matrix scale is M*N. Basic parallel algorithm ideas. Matrix A is stored on SM and matrices B and C are stored on VM. The row data of the A matrix is loaded scalar, the scalar is broadcast to the vector register, and the vector multiplies and accumulates with the column data of the B matrix, and the calculation result is saved as the value of the C matrix by rows[14]. Combined with the FT-MT2 core structure, on-chip capacity and instruction execution cycle, the matrix multiplied core-level code calculation scale is set to A matrix 6*512, B matrix 512*48, C matrix 6*48. According to the SIMD execution mode of 16 VPEs, the B matrix is divided in parallel with data, that is, the sub-matrix B' of 512*3 scale is processed on each VPE, and the sub-matrix C' of the target matrix of 6*3 scale is calculated[15]. The core code of matrix multiplication, benchmark time is 198.36 us. The vec_svbcast is to broadcast the A-matrix row data one by one to the vector registers. Three consecutive vec_mula are multiplied sequentially by the broadcast A matrix data and the 3 columns of the B matrix on each VPE, and accumulated to the target matrix C. Direction to be optimized: The inner loop cannot be expanded due to data dependence (tempC), because the inner loop completes a complete multiplication and accumulation process of row vectors and column vectors[16].

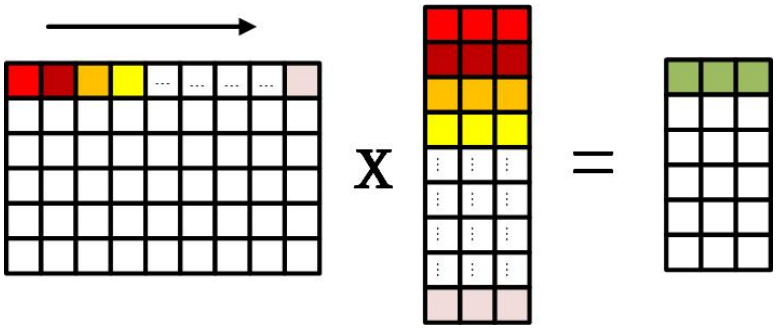


Figure 3. Schematic diagram of the matrix

```
void matrix(float* tempA, vector float* tempB, vector float* tempC)
{
    vector float temp1;
    int i;
    int j;

    for (i=0;i<6*3;i+=3){
        for (j=0;j<512*3;j+=3){
            temp1 = vec_svbcast(tempA[(i/3)*NumB+j/3]);
            tempC[i] = vec_mula(temp1, tempB[j], tempC[i]);
            tempC[i+1] = vec_mula(temp1, tempB[j+1], tempC[i+1]);
            tempC[i+2] = vec_mula(temp1, tempB[j+2], tempC[i+2]);
        }
    }
}
```

Figure 4. Schematic diagram of algorithm code

3. Matrix Multiplication Optimization

3.1 Code Refactoring and Loop Expansion

Code optimization (baseline execution time 159.40 us). Code refactoring: internal and external loop exchange, adjust matrix multiplication parallel algorithm. The A matrix is broadcast columnarly, multiplied by a row of the B matrix, and an additive update of all elements of the entire C matrix is completed[17]. Adding the compile option (-funroll-loops) allows the inner loop to fully expand the direction to be optimized. In the internal loop repeated fetch multiplication and addition statement, the 3MAC concurrency efficiency is low due to the coupling of memory fetch and computation.

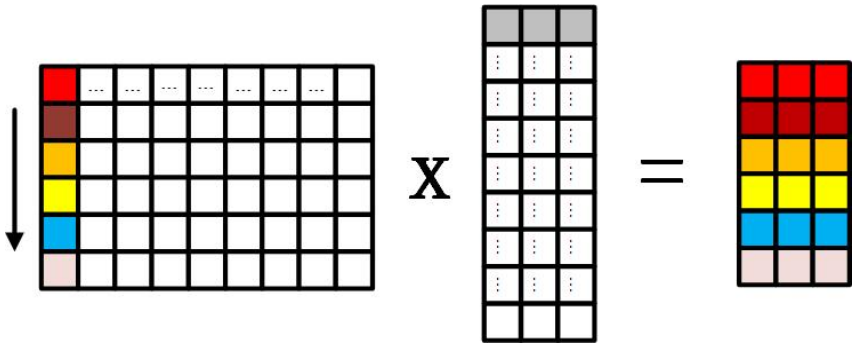


Figure 5. Schematic diagram of the matrix

```
void matrix(float* tempA, vector float* tempB, vector float* tempC)
{
    vector float temp1;
    int i;
    int j;
    int k;

    for (i=0;i<512*3;i+=3){
        for (j=0;j<6*3;j+=3){
            temp1 = vec_svbcast(tempA[(j/3)*512+i/3]);
            tempC[j] = vec_mula(temp1, tempB[i], tempC[j]);
            tempC[j+1] = vec_mula(temp1, tempB[i+1], tempC[j+1]);
            tempC[j+2] = vec_mula(temp1, tempB[i+2], tempC[j+2]);
        }
    }
}
```

Figure 6. Schematic diagram of algorithm code

3.2 Separation of Computation and Memory Access

Code optimization (baseline execution time 70.33 us). The B matrix merges the memory access, and the inner loop calculation is separated from the memory access. Direction to be optimized: Due to the multiple execution of the outer loop, the inner loop C matrix elements are repeatedly accessed and written[18].

```
void matrix(float* tempA, vector float* tempB, vector float* tempC)
{
    vector float temp1;
    int i;
    int j;
    int k;

    for (i=0;i<512*3;i+=3){
        k = i;
        vector float tmpb1, tmpb2, tmpb3;
        tmpb1 = tempB[k++];
        tmpb2 = tempB[k++];
        tmpb3 = tempB[k];
        for (j=0;j<6*3;j+=3){
            temp1 = vec_svbcast(tempA[(j/3)*512+i/3]);

            vector float tmpc1, tmpc2, tmpc3;
            k = j;
            tmpc1 = tempC[k++];
            tmpc2 = tempC[k++];
            tmpc3 = tempC[k];

            tmpc1 = vec_mula(temp1, tmpb1, tmpc1);
            tmpc2 = vec_mula(temp1, tmpb2, tmpc2);
            tmpc3 = vec_mula(temp1, tmpb3, tmpc3);

            k = j;
            tempC[k++] = tmpc1;
            tempC[k++] = tmpc2;
            tempC[k] = tmpc3;
        }
    }
}
```

Figure 7. Schematic diagram of algorithm code

3.3 Manual Deployment and Register Optimization

Code optimization (baseline execution time 36.12 us). The inner loop is fully manual unfolding[19]. Accumulate with registers instead of C matrices, and only one fetch and writeback of C matrices is performed outside the loop [20]. Direction to be optimized: The computational overhead caused by integer division can be optimized.

```
for (i=0;i<512*3;i+=3){
    k = i;
    l = i/3;
    vector float tmpb1, tmpb2, tmpb3;
    tmpb1 = tempB[k++];
    tmpb2 = tempB[k++];
    tmpb3 = tempB[k++];

    temp1 = vec_svbcast(tempA[l]);
    tmpc1_1 = vec_mula(temp1, tmpb1, tmpc1_1);
    tmpc1_2 = vec_mula(temp1, tmpb2, tmpc1_2);
    tmpc1_3 = vec_mula(temp1, tmpb3, tmpc1_3);

    temp2 = vec_svbcast(tempA[l+=512]);
    tmpc2_1 = vec_mula(temp2, tmpb1, tmpc2_1);
    tmpc2_2 = vec_mula(temp2, tmpb2, tmpc2_2);
    tmpc2_3 = vec_mula(temp2, tmpb3, tmpc2_3);

    temp3 = vec_svbcast(tempA[l+=512]);
    tmpc3_1 = vec_mula(temp3, tmpb1, tmpc3_1);
    tmpc3_2 = vec_mula(temp3, tmpb2, tmpc3_2);
    tmpc3_3 = vec_mula(temp3, tmpb3, tmpc3_3);

    temp4 = vec_svbcast(tempA[l+=512]);
    tmpc4_1 = vec_mula(temp4, tmpb1, tmpc4_1);
    tmpc4_2 = vec_mula(temp4, tmpb2, tmpc4_2);
    tmpc4_3 = vec_mula(temp4, tmpb3, tmpc4_3);

    temp5 = vec_svbcast(tempA[l+=512]);
    tmpc5_1 = vec_mula(temp5, tmpb1, tmpc5_1);
    tmpc5_2 = vec_mula(temp5, tmpb2, tmpc5_2);
    tmpc5_3 = vec_mula(temp5, tmpb3, tmpc5_3);

    temp6 = vec_svbcast(tempA[l+=512]);
    tmpc6_1 = vec_mula(temp6, tmpb1, tmpc6_1);
    tmpc6_2 = vec_mula(temp6, tmpb2, tmpc6_2);
    tmpc6_3 = vec_mula(temp6, tmpb3, tmpc6_3);
}
k = 0;
tempC[k++] = tmpc1_1;
tempC[k++] = tmpc1_2;
tempC[k++] = tmpc1_3;
tempC[k++] = tmpc2_1;
tempC[k++] = tmpc2_2;
tempC[k++] = tmpc2_3;
tempC[k++] = tmpc3_1;
tempC[k++] = tmpc3_2;
tempC[k++] = tmpc3_3;
tempC[k++] = tmpc4_1;
tempC[k++] = tmpc4_2;
tempC[k++] = tmpc4_3;
tempC[k++] = tmpc5_1;
tempC[k++] = tmpc5_2;
tempC[k++] = tmpc5_3;
tempC[k++] = tmpc6_1;
tempC[k++] = tmpc6_2;
tempC[k++] = tmpc6_3;
```

Figure 8. Schematic diagram of algorithm code

4. Conclusion

This paper studies open source software with good performance and frequent application in various fields, such as CFL3D and SU2 in fluid mechanics. NMSCC in scientific computing; GIMP in image processing; OCRE in acoustic optical signal processing, etc. Through the mastery of related tools such as Turbo C, Authorware, etc., understand how to build your own library in Turbo C, extend the

library in Authorware and other methods. Through continuous learning and innovation, we have developed a function library with superior performance for improving the basic function processing library of domestic processors. Based on the domestic DSP chip FT-M7002, C language and assembly language code are implemented and performance optimized for functions of basic algorithms such as matrix factorization, solving linear equations and filters. Huawei Kunpeng Server Project has developed image processing and signal processing function libraries such as HMPP on Huawei Kunpeng Server, and has obtained good research results.

References

[1] Cao Yang. Review of research and development status of domestic processors[J]. Computer Junkie, 2014 (11): 3. DOI: 10.3969/j.issn.1672-528X.2014.11.003.

[2] Several Policies to Encourage the Development of Software Industry and Integrated Circuit Industry.

[3] "Huawei Officially Releases Kunpeng and Ascend Crowdwisdom Program to Accelerate Basic Software Innovation".

[4] China Electronics officially released the "PK system" at the World Internet Conference[J]. Information Technology and Network Security,2018,37(1):3.

[5] SUN Liming. Optimization and implementation of computer graphics display system supporting domestic Feiteng 1500A processor[D]. Hunan: National University of Defense Technology,2016.

[6] YU Maoxue, JIA Dongning, WEI Zhiqiang, et al. A C++ Intelligent Source Code Conversion Framework Based on Domestic Heterogeneous Many-core Processor[J]. Computer Engineering and Science, 2021, 43 (6): 997-1005. DOI: 10.3969/j.issn.1007-130X.2021.06.007.

[7] LIU Xiaonan, ZHAO Rongcai, PANG Jianmin. Software Porting, Binary Translation and Domestic Processor Development[J]. Journal of Information Engineering University,2014,15(5):613-616,621. DOI: 10. 3969/ j. issn.1671-0673.2014.05.018.

[8] Huawei Developer Conference 2020: Arm Prosperity, Kunpeng Growth[J]. Science Chinese,2020(7):17-17.

[9] Kong Wen. ARM: Responding to the Post-PC Era with Industrial Chain and Ecological Environment[J]. Integrated Circuit Application, 2011(7): 20-21, 26.

[10] YU Wenping. ARM and China Unicom sign cooperation agreement to jointly build China's IoT ecosystem [J]. Internet of Things Technology,2019,9(3):3.

[11] SCLAFANI, ANTHONY J., DEHAAN, MARK A., VASSBERG, JOHN C., et al. Drag Prediction for the Common Research Model Using CFL3D and OVERFLOW[J]. Journal of aircraft,2014,51(4):1101-1117. DOI: 10.2514/1.C032571.

[12] CUI Qing, QIAN Xiaohui, LIU Jianming. Study on blunt body wingtip vortex based on open-source computational fluid dynamics software SU2[J]. Journal of Shenyang University of Aeronautics and Astronautics, 2020,37(3):24-32. DOI: 10.3969/j.issn.2095-1248.2020.03.003.

[13] BEUKES, MERVYN, HASTINGS, JOHN W. Self-Protection against Cell Wall Hydrolysis in Streptococcus milleri NMSCC 061 and Analysis of the Millericin B Operon[J]. 2002.

[14] MUELLER, ANDRE, VARGAS, EURIPEDES A., JR. Stability analysis of a slope under impact of a rock block using the generalized interpolation material point method (GIMP)[J]. Landslides,2019,16(4):751-764. DOI:10.1007/s10346-018-01131-1.

[15] CALLEBAUT I, MORNON JP. OCRE: a novel domain made of imperfect, aromatic-rich octamer repeats [J]. Bioinformatics,2005,21(6):699-702.

[16] YANG Xinfeng, LU Qingbin. Build your own library in Turbo C[J]. Journal of Microcomputer Applications, 2006,22(4):63-64. DOI: 10.3969/j.issn.1007-757X.2006.04.021.

[17] Research on parallel Laplace transform algorithm based on MPI and its application[D]. ZHANG Jiwei. Hubei University, 2011.

[18] Research and application of real-time attention monitoring brain-computer interface based on MPI parallel framework [D]. LI Yongchang. Lanzhou University, 2012.

[19] Research on parallel computing based on MPI[D]. ZHANG Zhihong. China University of Geosciences (Beijing), 2006.

[20] Preliminary implementation of MPI standard in distributed parallel computing[D]. HAO Lijie. Beijing

University of Chemical Technology, 2005.