Customized MMRF: Efficient Matrix Operations on SIMD Processors

Zhang Kai, Wang Yaohua, Chen Shuming, Li Zhentao, Wen Liang
School of Computer, National University of Defense Technology
Changsha 410073, China
Email: smchen@nudt.edu.cn

***Abstract*—Wireless communication and multimedia applications feature a large amount of matrix operations with different matrix size. These operations require accessing matrix in column order. This paper implements a Multi-Grained Matrix Register File (MMRF) that supports multi-grained parallel row-wise and column-wise access. We implement a 4*4 MIMO decoding with the help of MMRF to illustrate the efficient matrix operations on SIMD processors. Experimental results show that, compared with TMS320C64x+, our SIMD processor can achieve about 5.65x to 7.71x performance improvement by employing the MMRF. By customized design technology, we reduce the area and critical-path delay of MMRF by 17.9% and 39.1% respectively.**

Keywords- SIMD, Matrix Operations, Customize

INTRODUCTION *(HEADING 1)*

Wireless communication and multimedia applications, such as WiMAX, LTE and H.264, feature a large amount of matrix operations [1] [2] with different matrix size. These operations take up most of the execution time. Thus, the performance of these applications can be substantially improved by accelerating these matrix operations like the matrix multiplications and matrix transposes.

More recently, the Single Instruction and Multiple Data (SIMD) technique became commonly used, applying multiple parallel Processing Elements (PEs) to explore Data-Level Parallel (DLP) [5] by operating on multiple data in parallel in a single instruction. The performance of SIMD processors is restricted by the degree of data parallelism exploited from the program. In typical SIMD architectures, performance is lost when the PEs are not fully utilized or there are many data rearrangement operations among PEs.

Mapping matrix operations on SIMD processors brings a large amount of data rearrangement that decreases the system performance. These data rearrangement operations results from column-wise data accesses in matrix multiplications and matrix transposes. Take the matrix multiplication as an example: each matrix multiplication is processed by multiple PEs when implementing it on SIMD processors. The multiple PEs perform a dot operation on two vector data and then produce a scalar element. One of the two vectors is accessed in column-wise. However, the matrix data is usually organized by row-wise. Thus data rearrangement operations must be performer among multiple PEs.

To address this problem, Corbal proposed a novel matrix-oriented architecture MOM [3]. But MOM requires a much bigger multimedia register file and is mainly efficient for matrix add. Shahbahrami proposed the MRF [4] that provides both row-wise and column-wise access. But it is inflexible and inefficient when the matrix size is not consistent with the size of MRF. A polymorphic register file (PRF) [8] provides the system programmer with powerful means to organize the register file efficiently. The programmer can define the PRF by the matrix size. However, the PRF is difficult to implement due to the limit of instruction code. We proposed a CMRF [9] that supports multi-grained parallel row-wise and column-wise accesses in our previous work. The CMRF can achieve a significant performance speedup for matrix multiplications and matrix transposes.

This paper introduces a customized Multi-Grained Matrix Register File (MMRF) aimed at explaining the detail of parallel access modes more clearly. More important, we show more details about how to use the MMRF and make a customized design for MMRF.

MMRF ARCHITECTURE

As shown in Fig.1, the micro-architecture of the MMRF consists of a register array (RA), multiple read/write ports (R/WPs) and an index decoding logic. The RA consists of N*N register cells. The RA provides N row-vector registers (VR) VR_0 ~VR_{N-1} and N column-vector registers (CVR) CVR_0 ~CVR_{N-1}. Thus, the MMRF provides 2N vector registers logically by employing N^2 registers, which is two times of that of the traditional VRF.

The multiple R/WPs support parallel accesses to the MMRF, which can increase the parallelism degree of FUs. The BMR, which is configurable dynamically, is used to record the parallel access mode of the MMRF. The index view of the register varies as the content of BMR changes.

We can apply the MMRF to a typical SIMD processor without modification to their ISA. Generally, there is m-bit in an instruction word to encode a source or destination register index. The highest bit of the m-bit is used to distinguish the VR from the CVR. The other bits are used to indicate the indexed vector register in the VR or the CVR.

MULTI-GRAINED PARALLEL ACCESS

To demonstrate the concept of the MMRF, we implement a 16*16 MMRF on a SIMD processor with 16 PEs. We investigate several typical applications from wireless communication and multimedia area, and find out that the following three access modes are highly efficient:

A. One Way Access

In the one way access mode, the MMRF can be recognized as a traditional MRF. One VR or CVR, which is as shown in Fig.1, can be accessed in parallel by 16 PEs.

Published by Atlantis Press, Paris, France.

Generally, the one way access mode is mainly used to serve the 16*16 matrix. Programmers can access all the elements of one matrix row or column with the help of MMRF.

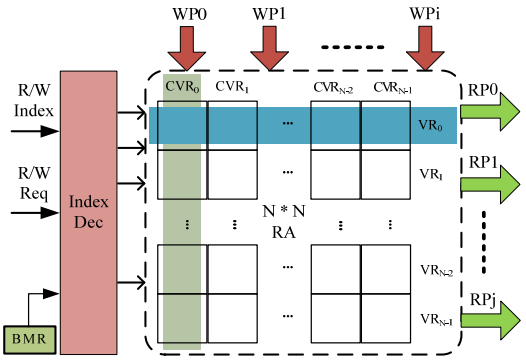

Figure 1. The micro-architecture of MMRF

B. *Two Ways Parallel Access*

The two ways parallel access mode is mainly used to serve 8*8 matrices. As shown in Fig. 2 (a), each VR or CVR consists of two parts from two different matrices. Each part can be accessed by 8 PEs. A row vector register like VR_i is the combination of VR_{i_a} and VR_{i_b}. A column vector register like CVR_j is the combination of CVR_{j_a} and CVR_{j_b}. Thus, two sub-VRs or two sub-CVRs can be accessed in parallel.

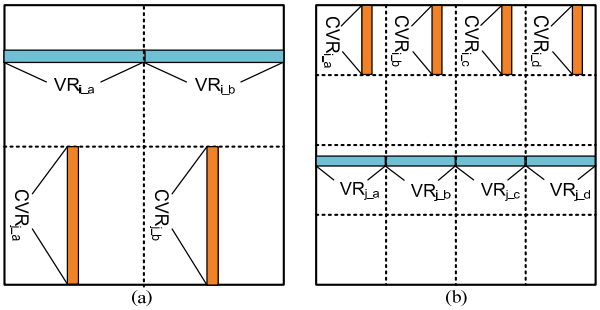

Figure 2. The parallel access mode of MMRF.

C. *Four Ways Parallel Access*

The four ways parallel access mode is mainly used to serve 4*4 matrices. As shown in Fig.2 (b), each VR or CVR consists of four parts from four different matrices. Each part can be accessed by 4 PEs. A row vector register VR_j is the combination of VR_{j_a}, VR_{j_b}, VR_{j_c} and VR_{j_d}. A column vector register CVR_i is the combination of CVR_{i_a}, CVR_{i_b}, CVR_{i_c} and CVR_{i_d}. Thus, four sub-VRs or four sub-CVRs can be accessed in parallel. Then four matrix multiplications can be performed in parallel with the help of MMRF. Fig.3 shows part of the 4*4 matrix multiplications process.

D. *Implementation of 4*4 MIMO decoding with MMRF*

Recently, Multiple-Input and Multiple-Output (MIMO) technologies which can significantly enhance the system performance have been adopted by the latest wireless standards such as WiMAX and 3GPP LTE. Taking a 4×4 (four sending antennas and four receiving antennas) MIMO-OFDM decoding process as an example.

There are two main methods used in linear MIMO decoder: zero-force (ZF) and minimum mean square Error (MMSE). Eq. (1) represents the ZF method. MMSE decoding is described in Eq. (2), where σ^2 is the noise variance.

$$\hat{S} = (H^H H)^{-1} H^H r \tag{1}$$

$$\hat{S} = (H^H H + \sigma^2 I)^{-1} H^H r \tag{2}$$

We configure the MMRF into four ways access mode to perform matrix multiplications in Eq. (1) and Eq. (2). Fig.4 shows a part of the assembly code that performs the computing process. The register BaseR0 and BaseR1 is the base address of matrix H^H and H respectively. The register BaseR2 is the base address of the result of H^H*H. The CMUL instruction performs a complex multiplication. The VREDUC4 instruction performs the add operation among the 4 PEs. After four VREDUC4 instructions, we get the first row of the H^H*H in VR12. Obviously, it shows that the MMRF can enable very convenient programming.

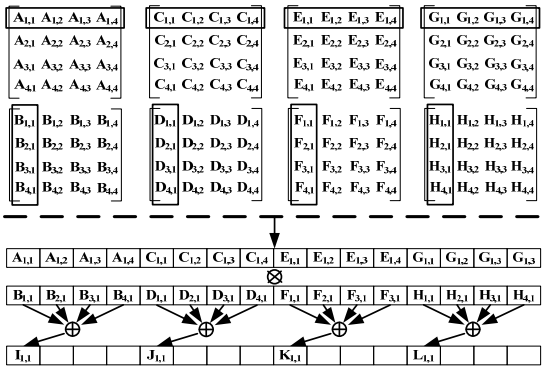

Figure 3. Part of Matrix Multiplications with MMRF

EXPERIMENTAL EVALUATION

We implement the MMRF in FT-Matrix-Sim which is a cycle-accurate simulator of FT-Matrix processor aimed at media applications. The FT-Matrix which includes 16 PEs is based on VLIW and SIMD technologies. The FT-Matrix accommodates 5-way VLIW instructions with two slots for LD/ST operations and one slot for MAC operations that support 32-bit complex multiplication.

We also implemented a traditional 16*16 VRF and a 16*16 MRF on the FT-Matrix-Sim. We compare the performance of the FT-Matrix that implements the VRF, the MRF, and the MMRF separately with that of a media processor core TMS320C64x+ (also reffered as C64x+). We select a set of typical algorithm kernels as benchmarks from wireless communication and multimedia applications.

Published by Atlantis Press, Paris, France.

E. *Experimental results*

Fig.5 shows the speedup of the FT-Matrix with the VRF, the MRF, or the MMRF over the C64x+. The maximal speedup of the FT-Matrix over the C64x+ should be 8 in theory by comparing their hardware resource. In Fig.5, the FT-Matrix with the MMRF can achieve nearly optimal speedup of 5.65x ~ 7.71x, as compared with the C64x+.

Meanwhile, the FT-Matrix with the MMRF achieves an average and maximal speedup of 2.21x and 5.87x respectively over the FT-Matrix with the VRF. And, the FT-Matrix with the MMRF achieves an average and maximal speedup of 1.6x and 2.22x respectively, as compared with that of the FT-Matrix with the MRF.

```
    MOV     #0002H,          BMR
    VLD     *BaseR0++[4],    VR0
||  VLD     *BaseR1++[4],    VR4
    VLD     *BaseR0++[4],    VR1
||  VLD     *BaseR1++[4],    VR5
    VLD     *BaseR0++[4],    VR2
||  VLD     *BaseR1++[4],    VR6
    VLD     *BaseR0++[4],    VR3
||  VLD     *BaseR1++[4],    VR7
    CMUL    VR0,   CVR4,     VR8
    CMUL    VR0,   CVR5,     VR9
    CMUL    VR0,   CVR6,     VR10
    CMUL    VR0,   CVR7,     VR11
    VREDUC4 VR8,   0,        VR12
    VREDUC4 VR8,   0,        VR12
    VREDUC4 VR8,   0,        VR12
    VREDUC4 VR8,   0,        VR12
    VST     VR12,  *BaseR2++[4]
```

Figure 4. Part of assembly code for 4*4 MIMO decoding

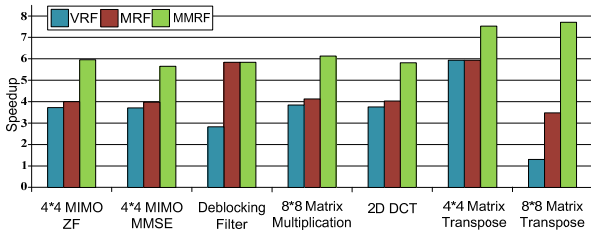

Figure 5. The speedup of FT-Matrix with the MMRF over C64x+.

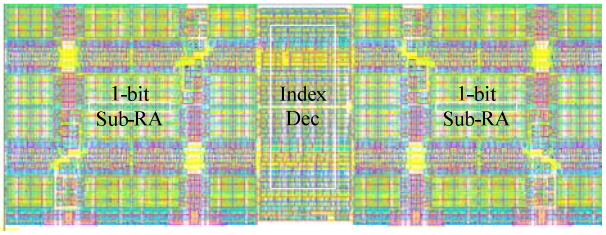

Figure 6. The layout of 2-bit sub-MMRF.

F. *Customized Design*

The MMRF with 3 read ports and 2 write ports has been implemented in VHDL. Each RP/WP can access the MMRF in row and in column. The RTL implementation has been synthesized with Synopsys Design Complier (DC) under TSMC 65nm technology. The placement and routing is executed by Encounter. The hardware cost after placement and routing is shown in Table 1. The critical-path delay is 1.6ns.

TABLE I. THE HARDWARE COST OF VRF, MRF AND MMRF

Component	area(mm2)	power(mW)
MMRF	0.56	94
MRF	0.54	92
VRF	0.49	82
FT-Matrix	15.88	1555

The register file always impacts the clock frequency of full chip and takes up significant on-chip area. To reduce the area and critical-path delay of MMRF, a hierarchical customized design technology is adopted on designing the RA and the index decoder of the MMRF. The whole MMRF is divided into 16 macro 2-bit sub-MMRF, and each macro 2-bit sub-MMRF, which can support row-wise and column-wise access, is composed of 2 1-bit sub-RAs. In addition, the 2 1-bit sub-RAs in a macro share a group of index decoder to reduce the area and power.

The hierarchical strategy is also employed for designing the read ports and write ports of the MMRF to obtain a compact layout and optimal wire interconnect. Each of the sub-MMRF contains 6 read ports and 3 write ports, including 2 write ports for row-wise access, 1 write port for column-wise access, 3 read ports for row-wise access, and 3 read ports for column-wise access as well. For one port, regardless of read or write port, it has a 1-bit sub-array of 16 word-lines by 16 bit. The 1-bit sub-array is partitioned into 4 4x4 small blocks. Each 4x4 block has its independently local word-line and shared global word-line. As a sequence, an extremely efficient layout and regular routing are achieved.

The layout of 2-bit sub-MMRF is shown in Fig.6. The layout area of MMRF is 0.46mm^2 under TSMC 65nm technology, exhibiting 17.9% reduction to the DC synthesized circuits. Furthermore, the customized MMRF achieves better speed performance. The critical-path delay of customized MMRF is 1.15ns. Compared with that of the DC synthesized circuits, the critical-path delay is reduced by 39.1%.

CONCLUSIONS AND FUTURE WORK

This paper introduces the MMRF, which can be well applied to existing SIMD processors without modification to their ISA, aimed at efficient matrix operations on SIMD processors. With the MMRF, we can obtain a performance speedup about 5.65x to 7.71x over TMS320C64x+ for the selected set of algorithm kernels from wireless communication and multimedia applications. By customized design technology, the area and critical-path delay of the MMRF reduced by 17.9% and 39.1%% respectively.

Published by Atlantis Press, Paris, France.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No.60906014 and 61070036) and Science Foundation of HPC Advisors Union.

REFERENCES

[1] Samsung. Downlink MIMO for EUTRA.3GPP TSG RAN WG1 meeting #44, 2006. 3GPP R1-060335.

[2] J. Andrews, A. Ghosh, R. Muhamed, "Fundamentals of WiMAX: Understanding Broadband Wireless Networking," Prentice Hall, Mar. 2007.

[3] Jesus Corbal, Roger Espasa, and Mateo Valero, "MOM: a Matrix SIMD Instruction Set Architecture for Multimedia Applications," In Proceedings of the ACM/IEEE SC99 Conference, pp. 1–12, 1999.

[4] Asadollah Shahbahrami, Ben Juurlink, and Stamatis Vassiliadis, "Versatility of Extended Subwords and the Matrix Register File," ACM Transactions on Architecture and Code Optimization, Vol. 5, No. 1, Article 5, Publication date: May. 2008.

[5] Mark Woh, Sangwon Seo, Scott Mahlke, Trevor Mudge, Chaitali Chakrabarti and Krisztian Flautner, "AnySP: Anytime Anywhere Anyway Signal Processing," ISCA'09, June 20–24, 2009.

[6] Brian Flachs, Shigehiro Asano, Sang H.Dhong, et al, "The Microarchitecture of the Synergistic Processor for a Cell Processor," IEEE Journal of Solid-State Circuits, Vol. 41, NO. 1, Jan.2006.

[7] Ronny Krashinsky et al, "The Vector-Thread Architecture," In Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004., pp.52-63, Jun.2004.

[8] Catalin Ciobanu, Georgi Kuzmanov, Georgi Gaydadjiev, Alex Ramirez, "A Polymorphic Register File for Matrix Operations," International Conference on Embedded Systems:Architectures, Modeling and Simulation, July. 2006.

[9] Kai Zhang, Shuming Chen, Hu Chen, Yaohua Wang, Xiaowen Chen, Sheng Liu and Wei Liu, "CMRF: a Configurable Matrix Register File for accelerating matrix operations on SIMD processors", *IEICE Electron. Express*, Vol. 9, No. 4, pp.283-289, (2012) .

Published by Atlantis Press, Paris, France.