International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.5 (2015), pp.343-354
http://dx.doi.org/10.14257/ijmue.2015.10.5.32

A 32nm EGPU Parallel Multiprocessor Based on Co-issue and Multi-Dimensional Parallelism Architecture

Yang Wang[1], Li Zhou[1*], Tao Sun[2], Yanhu Chen[1], Jia Wang[1], Yuanzhi Zhang[1], and Yuanyuan Gao[1]

[1]School of Information Science and Engineering, Shandong University, P.R. China [2]Shandong Provincial Key Laboratory of Network Based Intelligent Computing
University of Jinan, P.R. China
*vincent.sdu09@gmail.com, * zhou_li@sdu.edu.cn, Corresponding Author*

Abstract

In this paper, a Parallel Multiprocessor (PM) based on SIMT (Single Instruction and Multiple Threads) architecture is proposed. With co-issue architecture and multi-dimensional parallelism implemented in high-effective PM, Embedded Graphics Processing Unit (EGPU) provides great performance for various situations, such as general purpose computing, 3D scene rendering, and graphics processing. Application programs are departed into separated threads. Allocated by Thread Processing Unit (TPU), separated threads can be executed in parallel. Parallelism in different hierarchy and dimension are implemented by Multi-Dimensional Parallelism Processor (MD-PP), which has made a proper trade-off between performance and cost. Additionally, PM improves the hardware occupancy with its co-issue architecture and internal bus accessing mechanism to meet the demand of processing capability. Its unified shading architecture also helps to hide processing latency. PM can execute 4 basic operations in the best case and 2 in the worst case within each clock cycle. With 32nm process technology and 200MHz clock frequency, PM's area is about 5104494um^2, power consumption is about 101.838mW, and it can process nearly 28M vertices or fragments in average. Experimental results show that the MD-PP based PM can process data with high performance and get a balance between efficiency and hardware consumption simultaneously.

Keywords: *Embedded GPU, SIMT, Co-issue, Multi-Dimensional parallelism*

1. Introduction

Embedded application systems, such as mobile phones, hand-held consumer electronics and automobile electronics, have become more and more powerful and popular. People tend to rely on embedded systems, playing games with smartphones, watching high quality videos and images, editing photos, etc. Most of these functions are involved with powerful computation capability and complex data processing. That is the assigned task of EGPU processors. EGPUs have already become an essential component in current embedded systems [1-2].

GPU, first designed by NVDIA in 1999, is the most pervasive parallel processor to date [3]. GPUs have been widely used in both acceleration of 3D graphics processing and general purpose computing [4-7]. Along with the coming of big data age, how to improve the efficiency of mass data processing has become a challenging problem, which needs to be solved urgently by hardware systems. The efficiency of processing units has been more and more important in the design of modern GPUs, especially in embedded systems.

During the last decade, hardware design of GPU has been dramatically changed from fixed pipeline to unified programmable architecture, which leads to more flexible and efficient GPUs. However, unlike desktop GPU, EGPU design is limited by some critical factors, such as power consumption, area efficiency, and chip cost [8]. Most EGPUs still use traditional pipeline to do the graphics processing. How to enhance the performance of EGPU under limited condition is a critical challenge, which needs to be considered from both architecture innovation and technical integration.

Among all the challenges, the design of an efficient parallel architecture is the most critical one. There are several methods to implement the parallel computation, in which the most popular methods are SIMD (Single Instruction and Multiple Data) and SIMT (Single Instruction and Multiple Threads). Nowadays, SMT (Simultaneous Multithreading) has been proposed for better performance. In SIMD, elements of short vectors are processed in parallel. While in SMT, multiple instructions are issued from multiple threads and run in parallel. SIMT is somewhere in between: an interesting hybrid between vector processing and hardware threading [9]. Hence, the SIMT is the most promising model in parallel computation of GPUs, especially for EGPUs [10].

In this paper, a neoteric SIMT PM architecture with TPU and MD-PP is proposed. MD-PP allows multi-dimensional and hierarchical parallelism. Multi-threads are well mapped into multiple cores with the help of TPU. TPU reads instructions from Instruction Pool (IP) and allocates different threads to MD-PP. MD-PP consists of 4 Streaming Processors (SP). After all the data have been processed, MD-PP writes results back. Instruction operations include basic and complex calculations. Basic calculation refers to add, multiply, comparison, etc., which can be executed in 1 cycle. Complex calculation refers to sine, cosine, logarithm, trigonometric, etc., which need multiple cycles. By separating complex calculations from basic calculations, SPs only deals with basic operations. Complex calculations are processed by Special Function Processor (SFP). There are 4 SP cores and 1 SFP core integrated in a PM. Five PEs in each SP core are organized in a hierarchy way, which can be configured to work in pipeline or parallel way. Based on MD-PP architecture, software threads can be mapped to MD-PP and SFP, which unifies the software programming and hardware execution. And it will bring more convenience for EGPU programming.

In this work, the following contributions have been made:

1) TPU is designed to dispatch separated threads to deal with corresponding data based on the same instructions. TPU balances the workload of execution cores dynamically and efficiently.
2) The architecture of MD-PP is designed to work simultaneously. MD-PP achieves multi-dimensional parallel processing in both horizontal and vertical direction.
3) Based on the co-issue architecture, instructions are combined according to parallelism dimension, and hardware utilization is also increased.
4) An efficient anti-collision bus accessing mechanism is applied in PM to solve the conflicts between different executing units. Memory accessing latency can be hidden dramatically.

The rest of paper is organized as following. In Section 2, the development of GPU and EGPU architecture is introduced. Section 3 describes the detail architecture design of proposed PM. Experimental results are provided in Section 4. Section 5 presents the conclusion and further consideration of future research works.

2. Development of GPUs with SIMT Architecture

Modern GPU has evolved from a traditional fixed pipeline to a programmable parallel processor with computing capability exceeding that of multicore CPUs [8]. Comparing with the traditional pipeline, which can only deal with graphics tasks, programmable processor is more flexible. With the advent of vertex shader and fragment shader, graphics processing is divided into vertex stage and fragment stage. Vertex stage works on the vertices of primitives, including points, lines, and triangles. Typical operations transform coordinates into screen space before setting up and raster. And lighting and texture parameters are also prepared for fragment processing. Fragment stage fills the interior of primitives for raster output, along with the interpolated parameters [8].

However, the workloads of vertex and fragment stages are not well balanced sometime in different situations. So unified processor architecture is presented, which can execute both vertex and fragment programs. Unification enables dynamic load balancing of varying vertex processing and fragment processing workloads, and permits the new graphics shader stage, geometry stage [11-13]. The generality required of a unified processor opened the door to a completely new GPU parallel-computing capability.

To further accelerate data processing, some GPU parallel processing architectures are presented based on the unified architecture. SIMT, which has been widely used, is a good choice for GPU and EGPU architecture design. NVDIA SIMT hardware architectures provide a convenient platform for CUDA (Compute Unified Device Architecture) programming [3]. CUDA’s hierarchical threads map to hierarchical processors of SIMT GPUs. GPU executes one or more CUDA kernel grids, where a PM core in GPU executes one or more thread blocks; several SP cores and other execution units in PM complete threads execution [3]. Once the instruction is set, multiple registers are set, multiple addresses, and multiple flow paths are ready for execution. This kind of SIMT architecture brings great advantages in software programming flexibility and execution efficiency. With the development of mobile technologies, parallel processing technology has been massively adopted in EGPUs.

3. MD-PP based Parallel Multiprocessor

In this section, MD-PP based PM architecture is presented in detail. PM consists of 5 parts, IP, TPU, MD-PP, SFP, and internal memory, as shown in Figure1. Command Processor (CP), Data Preparation Buffer (DPB) and Data Output Buffer (DOB) are also demonstrated in the diagram. IP, with a depth of 32, is designed to store the instructions from CP. TPU reads 4 instructions each time from IP and packs them as a warp, which is then allocated to MD-PP as independent threads. Specifically, TPU allocates an instruction package, including 4 independent instructions, to MD-PP’s execution units to achieve 4 co-issues at one time. MD-PP is organized to deal with basic calculations which can be finished within one clock cycle. SFP focuses on complex calculations, which may need multiple cycles. Memory unit is a combination of memory blocks used for various purposes. Constant buffer contains constant used by MD-PP and SFP in EGPU’s shading stage. Shared memory is commonly owned by both MD-PP and SFP. To resolve data interaction and reusing problems during graphics processing, shared memory is designed to store intermediate results. It saves operating time, and finally accelerates processing. Besides texture, relative memory also stores data for shading stages.

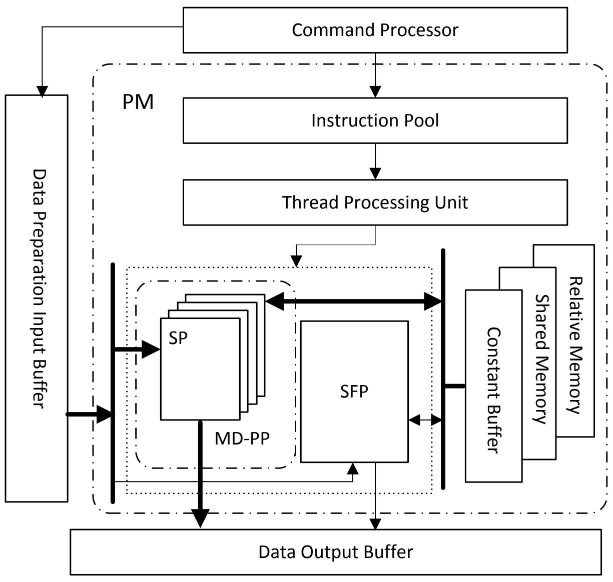


Figure 1. Overview of PM

PM processing begins with the commands from CP. CP communicates with host CPU, responds to the command of CPU, and coordinates all the blocks of EGPU. CP fetches the instructions from system bus. After analyzing each instruction, CP sends control signals to the relative blocks, awaking them to execute the current instruction. Since this paper focus on MD－PP design, the details of CP and data preparation process are not included. Once instructions are pushed into IP, PM is awoken and starts to run. PM executes 4 instructions at the same time with the help of co-issue mechanism in TPU. By sharing the results of each SP in shared memory, the speed of calculation is dramatically increased. When the current instruction is finished, result will be written back to DOB. If IP's empty state is detected, PM goes into idle state, and waits to be awoken by new instructions from CP.

3.1. Instruction Design

PM instruction is 64-bit SIMT instruction set to implement various kinds of operations, as shown in Table 1. Besides fundamental operation code, destination operand, and source operands, specific bits are designed to distinguish different threads, dimensions, and operation types.

Table 1. Format of Instruction

[63:60]	[59:57]	[56:49]	[48:41]	[40:33]
Op-code	Type	Destination	Scr1	Scr2
[32:25]	[24:23]	[22:21]	[20:17]	[16:0]
Scr3	Dimension-tag	Thread-id	Synchronization	Reserved

Different operations are distinguished by 3-bit type code, which is used to separate the basic operations from complex ones, and to indicate specific types of operations, such as algorithm operations, logic operations, or memory access. Cooperated with 4-bit op-code, the instruction can provide up to 16 kinds of specific operations for each instruction type. That can cover most of the operations of EGPU's processing.

Four dimension types are designed, 1D~4D. All operations are classified according to their dimension types. Individual SP is 1D scalar processor, dealing with 32-bit floating operations. So 64-bit operations, 96-bit operations, and 128-bit operations are distributed as 2D, 3D, and 4D thread operations. Two-bit dimension tag in instruction is used to mark the dimension of operations, which is an important factor for computing efficiency.

Threads and SP cores are mapped based on MD-PP architecture. They are distributed to different SPs according to their thread ID. Two-bit thread ID is set to signify 4 threads. Another 4 bits are set as synchronization bit. The rest bits of the instruction are reserved for further extensions.

3.2. Thread Processing Unit

SP is a 32-bit scalar processor. To deal with common 128-bit operations in graphics processing, 4-channel should be used simultaneously. For example, coordinates transformation in vertex processing and the colors conversion in fragment processing are all 128-bit operations. Instructions are expected to execute in 4 SPs at the same time. So instructions must be assigned into different threads, and mapped to different SPs. TPU, as shown in Figure 2, is designed to distribute threads. TPU reads instructions from IP, packs several instructions together and then sends to MD-PP. It guarantees the utilization of hardware according to their dimension tags. When MD-PP is idle, TPU distributes the corresponding instruction packages. There are mainly 4 different cases, which can be processed in parallel. 4D instructions, referring to 128-bit operations, can be assigned into 4 independent SPs as different threads. Each SP channel deals with its 32-bit data according to the thread ID. For 3D and 2D instructions, MD-PP can’t be fully occupied. So these instructions should be co-issued with others to fully utilize MD-PP. 3D and 1D instruction are combined to launch simultaneously. Three SPs are used to deal with 3D instruction, and one SP processes 1D instruction simultaneously. Similarly, two 2D instructions are packaged together. Four independent 1D instructions are allocated to 4 SPs in MD-PP. TPU receives acknowledge signals from SPs, and balances the workload dynamically.

3.3. MD-PP and SFU

Since graphics operations include basic operations and complex operations, the processing units are organized as MD-PP and SFP. Considering the complexities of SFP operations in accessing Look-Up Table (LUT), SFP unit is separated from MD-PP and shared by the SPs to reduce the consumption of area on chip.

MD-PP is designed to achieve the multi-dimensional parallelism, in direction of both horizontal and vertical. The horizontal parallelism is realized by utilizing 5 independent PEs in each SP as execution pipeline. Vertical parallelism is realized by taking advantage of 4 SPs in MD-PP, operating in parallel to execute independent threads.

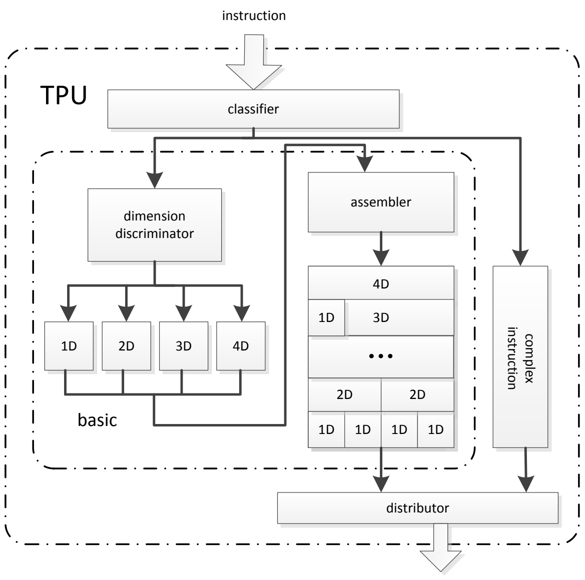


Figure 2. Overview of TPU

For each PE, execution pipeline is divided into 5 sub-stages: Instruction Fetch (IF), Instruction Decode (ID), Memory Access (MA), Execution (EXE), and Write Back (WB). Instruction is firstly fetched from instruction FIFO, and then decoded in ID stage. For algorithm or logic instruction, after reading relative data in MA stage and calculating the result in EXE sage, it writes those results back to the output data buffer in WB stage. When memory access instruction is executed, the relative address is calculated in EXE stage in advance to support memory accessing in MA stage. Sub-stage results are stored in General Purpose Registers (GPR) array. GPR array is the interface to access the shared memory and to communicate with other PEs. The internal architecture of SP is shown in Figure 3.

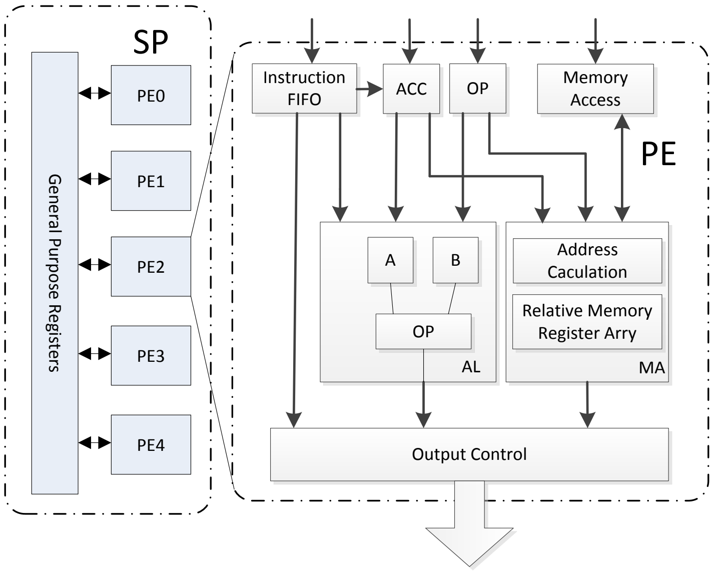


Figure 3. Architecture of SP

SFP, which is designed for complex calculation, is organized as a longer pipeline for intricate operation, as shown in Figure 4. SFP pipeline is similar to SP, except an additional LUT unit and execution stage. Three extra execution cycles are added to meet the requests of complex operations. Execution cycles are defined as EXE0, EXE1, EXE2, and EXE3. SFP execution core is called Computing Units (CU). Compared with PE cores which can only deal with basic operations, CU cores are mainly used to execute complex operations.

LUT uses an array indexing to get complex calculation result directly. Since retrieving a value from memory is often faster than undergoing a complex algorithm or logic computation, the execution time is reduced significantly. The input operand width determines LUT accuracy and hardware realization area. High accuracy leads to a bigger hardware logic area, which is one of the critical factors in embedded platforms. The input operand width of LUT in this paper is 6-bit width along with 3-bits for function control. Functions implemented by LUT include reciprocal, sine, cosine, exponent, and binary logarithm.

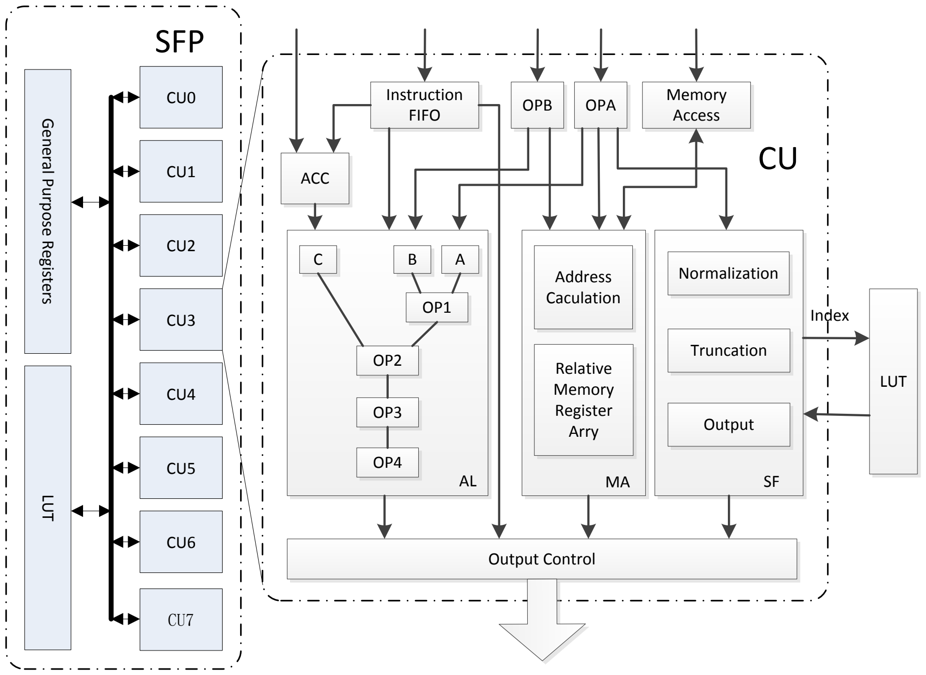


Figure 4. Architecture of SFP

Floating-point operations are implemented in both PE cores and CU cores. Compared with integer operations in the same bit length, floating-point numbers are more accurate and can support a wider range of values. Moreover, hardware implementation of floating-point arithmetic is also with high efficiency comparably with integer. Input integer can be converted to 32-bit floating-point format, and then be utilized in PE and CU cores.

3.4. Bus Accessing Mechanism and Memory Architecture

Different instructions are classified and repacked based on parallelism dimension. For 4D operations, data is organized in 128-bit, and can be read in 1 clock cycle, which leads that all the SPs executing simultaneously. This is the best parallel case based on MD-PP architecture. For 1D~3D dimension instructions, operations are packed with different data which cannot be ready synchronously, so the efficiency PM are limited by the memory accessing throughput. That is critical for EGPU execution performance.

Two AXI buses are added to guarantee data accessing efficiency of MD-PP. When 4D instructions are executed, 128-bit data will be read at one time. Another situation, 3D instruction combined with 1D one, 2 independent data (96-bit and 32-bit), are read simultaneously. Similarly, 2D and 2D case also needs to read data twice. Based on double

AXI buses, all data can be prepared in 1 clock cycle for all the above situations. For the worst case, which is the packed instruction consisting of 1D operations, memory access cycle can also be reduced to 2 clock cycles. For SFU, additional memory channel is reserved to speed SFU memory accessing. Table 2 shows the pipeline of PE and CU. It is obvious that only one PE occupies the memory bus of SP in each cycle. The executions of 5 PEs are stagger. Eight CUs are also well organized with the same strategy.

Table 2. Pipeline of PE and CU

IF	Instruction Fetch	ID	Instruction Decode	MA	Memory Access	
EXE	Execution	WB	Write Back			

Clock	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CU0	IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB
CU1		IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID	MA	EXE0	EXE1	EXE2	EXE3
CU2			IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID	MA	EXE0	EXE1	EXE2
CU3				IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID	MA	EXE0	EXE1
CU4					IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID	MA	EXE0
CU5						IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID	MA
CU6							IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF	ID
CU7								IF	ID	MA	EXE0	EXE1	EXE2	EXE3	WB	IF
PE0	IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB	IF
PE1		IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB
PE2			IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB	IF	ID	MA	EXE
PE3				IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB	IF	ID	MA
PE4					IF	ID	MA	EXE	WB	IF	ID	MA	EXE	WB	IF	ID

A flexible memory accessing architecture is proposed in this paper. PM contains 3 memory hierarchies, that is, external memory, internal memory, and FIFOs. Graphics source data and final processing results are stored in external memory DPB and DOB. Internal memory consists of 3 parts, constant buffer, shared memory, and internal function memory. Unlike the constant buffer and internal function memory, shared memory causes data conflicts inevitable. It is defined that writing accessing has higher priority than reading, then the conflictions of read and write can be prevented. This kind of conflicts may not be aware by software programmers can be solved by the hardware. Shared memory is separated into 5 independent banks corresponding to 4 SPs and SFP. Each PE can only write intermediate data into its own bank. So conflicts caused by different processing units writing to the same address can be avoided. If the same memory address is read by multiple cores, different priorities are distributed according to thread IDs. The same strategy is also applied into the constant buffer and internal function memory. FIFOs are used in PEs and CUs to store instructions temporarily. Details of designed internal memory are presented in Table 3 as following.

Table 3. Details about Internal Memory

Name	Description	Size
Constant Buffer	Store constant	4Kb, 128*32bit
Shared Memory	Share intermediate data	6Kb, 4*32*32bit + 64*32bit
Relative Memory	Store relative memory	8Kb
Single FIFO	Store instructions	1Kb, 16*64bit

With hierarchical memory architecture and flexible bus accessing mechanism, memory accessing time is greatly reduced and the efficiency of the whole PM is improved.

4. Experimental Results

In this section, experimental results are presented to evaluate the performance and the cost of the whole PM design.

A verification platform is established to verify the performance of the PM design proposed in this paper. Instructions are designed in different test cases to verify whether PM's function is correct. The relative data are prepared in external memory in advance. All instructions and data in test cases meet the design rules of PM. Verification results are given out after comparing with golden data.

Processing performance is another verification target. Instruction numbers executed per second can reflect the performance directly, but it relied on the amount hardware resources to a great extent. The increasing use of hardware resources increases area and power consumption. The balance between performance and hardware cost is considered during PM design. Clock cycle numbers spent by each instruction are used to evaluate the performance in this paper. Table 4 presents the result of processing.

Table 4. Design Indicators

Case Number	Instructions Package Type : Number (4 basic operation)	Clock Cycles	Execution Unit
1	1D*4 : X	X+5	MD-PP
2	1D+3D : X	X+4	MD-PP
3	2D+2D : X	X+4	MD-PP
4	4D : X	X+4	MD-PP
5	Complex Operation : X	X+7	SFP

Instruction is organized by basic 32-bit floating operation. The performance is evaluated by the capability to calculate basic 32-bit floating operations, and instruction package (from 1D to 4D) contains 4 basic operations. Based on parallel MD-PP architecture, the processing performance has been greatly improved. In execution sub-stages, PM can process 4 floating operations per cycle in the best cases (4D). In the worst cases (1D), 2 floating operations are finished in each cycle. With a large number of programed instructions, almost 4 basic operations can be executed in each clock cycle. At 200MHz frequency, about 28M vertices or fragments can be processed in average. In general, the PM has reached a preferable quality. PM's backend design is listed in Table 5 and Figure 5. The voltage of standard cells provided in the library ranges from 0.7V to 1.25V, while the temperature ranges from 25℃ to 125℃. Based on worst PVT corner, area, power, frequency and layout results are given out.

Table 5. Technology Condition & Results

Processing Technology	32nm CMOS
Voltage	0.7V
Temperature	125℃
Area	5104494um^2
Power Consumption	101.838mW
Frequency	200M

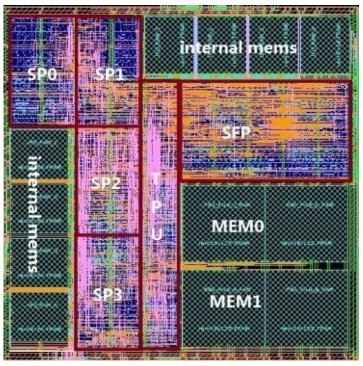


Figure 5. Layout of PM

5. Conclusion

With the limitation of embedded devices, the innovations of architecture and processing mechanism are playing more and more important roles in the design of embedded devices. This paper provides an exploration of parallel processing in embedded processor design. An efficient PM architecture is proposed with a co-issue and MD-PP technology. TPU dispatches multiple threads and MD-PP implements multi-dimensional parallelism. Specific bus accessing mechanism is designed to avoid memory accessing conflicts. PM can be used in both general purpose computing and graphics processing acceleration in order to meet the increasing requirements of data processing. Experimental results show that the PM can execute instructions efficiently, completing 4 basic operations in the best case and 2 in the worst case within each clock cycle. And it can process about 28M vertices or fragments per second with 32nm CMOS design technology library and 200M clock frequency. Its area is about 5104494um^2, and power consumption is nearly 101.838mW. Further improvements can be obtained by integrating more processing cores. The relationship between thread smart distribution and memory accessing confliction could also be further studied. How to get the balance between performance and cost is another long-term research topic in EGPU design. These research directions will be covered in the next stage of research.

Acknowledgements

This work was supported by Natural Science Foundation of Shandong Province (ZR2013FQ006), Fund of Independent Innovation in Shandong Province (2013CXB3020), Foundation Grant of State Key Laboratory of ASIC & System (10KF011), State Key Laboratory of Digital Multimedia Technology (2013-1-2569), Shandong Post-Doctor Innovation Foundation Grant (201002029), Shandong University Innovation Foundation Grant (10000059614021), Grant of China Post-Doctor Innovation Foundation (20110491601) and University of Jinan Ph.D Foundation Grant (xbs1018). The authors would like to thank all research partners for their significant contributions in this work. Thank Jinan University for supporting in hardware platform.

References

[1] L. Garber, “GPUs Go Mobile”, Computer, vol. 46, no. 2, (**2013**), pp. 16-19.
[2] Imagination Technologies Ltd, “POWERVR Series5 Graphics SGX Architecture Guide for Developers”, 1.0.8 Ed., Imagination Technologies Ltd, (**2011**).
[3] C. M. Wittenbrink, K. Emmett, and A. Prabhu, “Fermi GF100 GPU architecture”, IEEE Micro, vol. 31, no. 2, (**2011**), pp. 50-59.
[4] L. Santos, E. Magli, R. Vitulli, J. F. Lopez, and R. Sarmiento, “Highly-parallel GPU architecture for lossy hyperspectral image compression”, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 6, no. 2, (**2013**), pp. 670-681.

[5] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing", Proceedings of the IEEE, vol. 96, no. 5, (**2008**), pp. 879-899.
[6] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the future of parallel computing", IEEE Micro, vol. 31, no. 5, (**2011**), pp. 7-17.
[7] A. Bayoumi, M. Chu, Y. Hanafy, P. Harrell and G. Refai-Ahmed, "Scientific and engineering computing using ati stream technology", Computing in science & engineering, vol. 11, no. 6, (**2009**), pp. 92-97.
[8] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture", IEEE Micro, vol. 28, no. 2, (**2008**), pp. 39 - 55.
[9] Y. Kreinin, "SIMD< SIMT< SMT: parallelism in NVIDIA GPUs", (**2011**).
[10] H. Y. Kim, Y. J. Kim, J. H. Oh, L. S. Kim, "A Reconfigurable SIMT Processor for Mobile Ray Tracing with Contention Reduction in Shared Memory", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 4, (**2013**), pp. 938 - 950.
[11] Y. K. Lai, and Y. C. Chung, "3-D graphics processor unit with cost-effective rasterization using valid screen space region", IEEE Transactions on Consumer Electronics, vol. 59, no. 3, (**2013**), pp. 705 - 713.
[12] T. Schiffer, and D. Fellner, "Ray Tracing: Lessons Learned And Future Challenges", IEEE Potentials, vol. 32, no. 5, (**2013**), pp. 34 - 37.
[13] J. Spjut, A. Kensler, D. Kopta, and E. Brunvand, "TRaX: A Multicore Hardware Architecture for Real-Time Ray Tracing", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 12, (**2009**), pp. 1802 - 1815.
[14] S. F. Hsiao, P. H. Wu, C. S. Wen, and L. Y. Chen, "Design of a programmable vertex processor in OpenGL ES 2.0 mobile graphics processing units", 2013 International Symposium on VLSI Design, Automation, and Test (VLSI-DAT), IEEE, (**2013**).
[15] J. Balfour, W. J. Dally, D. Black-Schaffer, V. Parikh, and J. Park, "An Energy-Efficient Processor Architecture for Embedded Systems", Computer Architecture Letters, vol. 7, no. 1, (**2008**), pp. 29-32.
[16] Y. Wang, X. Zhou, L. Wang, J. Yan, W. Luk, C. Peng, and J. Tong, "SPREAD: A Streaming-Based Partially Reconfigurable Architecture and Programming Model", IEEE Trans. VLSI Syst., vol. 21, no. 12, (**2013**), pp. 2179-2192.
[17] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A. E. Lefohn, and T. J. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware", Computer graphics forum, vol. 26, no. 1, Blackwell Publishing Ltd, (**2007**).
[18] M. M. I. Bhuiyan, "Designing a Line-clipping Algorithm by Categorizing Line Dynamically and Using Intersection Point Method." 2009 International Conference on Electronic Computer Technology, IEEE, (**2009**).
[19] M. Pharr, and R. Fernando, "GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation", Addison-Wesley Professional, (**2005**).
[20] M. Daga, A. M. Aji, and W. Feng, "On the Efficacy of a Fused CPU+ GPU Processor (or APU) for Parallel Computing", 2011 Symposium on Application Accelerators in High-Performance Computing (SAAHPC), IEEE, (**2011**).

Authors

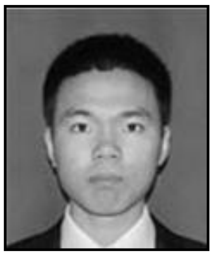
Yang Wang, he was born on May 10, 1990. He received the BEng degree in Integrated Circuit Design and Integrated System from Shandong University, China, in 2013, and now he is pursing the MEng degree in Circuit and System in Shandong University. His current research interests include 3D-GPU and Computer Architecture etc.

Li Zhou, she received doctor degree on 2004 from Zhejiang University, China. On 2004, she joined Freescale semiconductor R&D center as principle design architect till 2009. As an architect and core team member, she participated in multiple National high-tech SoC and HDTV fundamental research projects, led multiple 65nm/90nm 10 million gate scale VLSI SoC chips, and joined many VLSI project and architecture researches. From 2009, She is an assistant professor in Shandong University, China. Her current research interests include stereo vision system algorithm and hardware design, GPU architecture and VLSI design, high performance processor architecture and VLSI design, etc.

Tao Sun, he received doctor degree on 2002 from Zhejiang University, China. He joined Suzhou China Core Co. Ltd as a researcher on CPU architecture and vice-general manager till 2007. From 2007 to 2009, Dr. Sun was a senior manager in Spansion China, led VLSI design projects. From 2009 till now, he is an associated professor in University of Jinan, China. His research interests include CPU/VLSI architecture, Solid storage architecture, etc.

Yanhu Chen, he received doctor degree in 2007 from Chinese Academy of Sciences, China. Since 2007 he joined the School of Information Science and Engineering of Shandong University, where he worked as an Assistant professor in the institute of electronic design automation technology. His main present research interests are in the fields of compound semiconductor RF& Microwave device design, modeling and simulation technology; RFIC & MMIC design; other new structure and new material semiconductor device design, simulation and their integrated technology.

Jia Wang, she was born in October 1988, in China. She received the BEng degree in Integrated Circuit Design and Integrated System from Shandong University, China, in 2011, and now she is pursing the doctor degree in Electronic Engineering in City University, Hong Kong. Her current research interests include VLSI design, SoC design, and chips design for security.



Yuanzhi Zhang, he received bachelor degree of engineering on 2011 from Shandong University, China. From 2011 till now, Zhang is pursuing his master degree in Shandong University. His research interests include CPU/GPU architecture design, human intelligence, image processing, etc.

Yuanyuan Gao, she received bachelor degree of engineering on 2012 from Shandong University, China. From 2012 till now, Gao is pursuing her master degree in Shandong University. Her research interests include CPU/GPU architecture design, human intelligence, image processing, global illumination, etc.