A Design Space Exploration Method for on-Chip Memory System Based on Task Scheduling

Hongyu Meng[1 2], Pengfei Ding[3], Mingxuan Wang[3] and Donglin Wang[1]
[1]*Institute of Automation, Chinese Academy of Sciences, Beijing, China*
[2]*University of Chinese Academy of Sciences, Beijing, China*
[3]*Beijing LogicSmart Technology, Beijing, China*
menghongyu2014@ia.ac.cn, 1522253860@qq.com, xuanyaoxi@163.com, donglin.wang@ia.ac.cn

Abstract—**As technology scales in the past sixty years, System on-Chip (SoC) has become the inevitable direction of chip development. Multi-core processor as one of the SoCs is the most effective solution to balance the performance and power. However, the bandwidth of chip system has become the bottleneck for multi-core processor. Improving the bandwidth of off-chip memory system or introducing a well-designed on-chip memory system can solve this issue relatively. In this paper, we introduce a design space exploration method based on task scheduling to guide the design of on-chip memory system. Using this method, the utilization of cores can reach a high level under the benchmark of DGEMM and some hardware constraints. In addition, we conclude some relationships for the parameters of the multi-core architecture, which can instruct us how to design appropriate on-chip memory system.**

Keywords-design space exploration; multi-core architecture; memory system; task scheduling

I. INTRODUCTION

In 1965, Gordon Moore published an article predicting that the complexity for minimum component costs has increased at a rate of roughly a factor of two per year [1]. Then in 1975, Moore re-described his prediction that the number of transistors in integrated circuit would increase at an average rate of every 18 mouths, which was later called Moore's Law. After nearly half a century of facts, the development of integrated circuit is basically consistent with Moore's Law. Processor as one of integrated circuit has developed rapidly over the past few decades.

In the 1950s and 1960s, processors were improved by the use of carry look-ahead adder, booth algorithm and Wallace tree. Then in the 1970s, Instruction Set Architecture (ISA) gradually became the mainstream research direction, and pipeline and multiple issue structure have been used in processors designs which can increased the performance while maintaining a serial programming model. During the next several decades especially after the appearance of Reduced Instruction Set Computer (RISC), the development of instruction level parallelism reaches its peak. With the advent of Intel's Pentium 4 processor using the third-generation Prescott core, the drawbacks of the traditional single-core processor began to emerge such as the excessive power dissipation which could prevent the processor from working properly. Then in 2006 Intel Developer Forum (IDF), Energy Efficient Performance was proposed and performance per watt was the focus of future processor development. The multi-core structure utilizes Multiple Instruction Multiple Data (MIMD) and arranges several low-level and low-power cores on one chip to offer higher performance, better parallelism and low power consumption [2]. That means multi-core structure can better balance the performance and power, and therefore multi-core processor has become the mainstream of processor design. However, with the development of multi-core processor, some issues started to crop up such as Programming Wall and Bandwidth Wall. Improving memory bandwidth and introducing shared on-chip memory can both solve the Bandwidth Wall.

The memory in processor system can be mainly divided into two categories: Dynamic Random Access Memory (DRAM) and Static Random Access Memory (SRAM). The traditional DRAM is only composed of one transistor and one tiny capacitor. Due to the small area and high integration level, DRAM can be used for large-scale data storage [3]. Though the latency of DRAM is high, by integrating the synchronizer and pre-fetch module, DRAM can be used to customize Synchronous DRAM (SDRAM) and Double Data Rate SDRAM (DDR SDRAM, DDR for short) which can provide high-speed data. In recent years, major storage vendors have been researching and upgrading DDR SDRAM, and DDR SDRAM developed to the fourth generation (DDR4 for short). Each data line in a DDR4 chip can deliver data rate up to 3200 Mbps which can meet single-core processor but not multi-core processor. Thus some chipmakers began to use Graphic Double Data Rate Synchronous Graphic Random Access Memory (GDDR SGRAM, GDDR for short) and High Bandwidth Memory (HBM) to solve the problem of bandwidth for multi-core chip system. Compared with DDR, the intention of the traditional GDDR is for graphics cards so that the amount of prefetching data can be larger which means the data rate can be higher. For the GDDR that has been used in the industry, the rate of each data line can reached 16 Gbps. Unlike GDDR, which improves higher bandwidth by increasing the rate of each single line, HBM improves the bandwidth by increasing the amount of data lines. Usually a stack of HBM integrates 4 DRAM chip, each with 256-bit data width, which is 16 times that of conventional DDR. In order to reduce power consumption, the prefetching data is only a quarter of DDR, and the data rate of each data line in HBM is 2 Gbps. But because of its integration of multiple high-width DRAM, a

single HBM stack's bandwidth can reach 2048 Gbps. In addition, HBM adopts 2.5D packaging to reduce the board-level area, and HBM is packaged with the chip together using silicon interposer to reduce the pins of the packaged chip. Although the bandwidth of the off-chip memory system can be improved in many ways, the bandwidth limitation of the on-chip interconnection is still a bottleneck to the Bandwidth Wall. For computing applications with large memory access ratios, introducing on-chip memory system is also an effective solution. Traditional on-chip memory system mostly consists of SRAM. SRAM consists of 6 transistors and does not need any additional module to control it. Though the area of SRAM is larger than DRAM, its latency is very low. Thus for on-chip memory system, SRAM is a batter a selection [4]. By introducing on-chip memory, the need for throughput of off-chip memory system can be greatly slowed down while can also improve the parallelism of multi-core structure. However, there is no reasonable solution at present which can guide the design of on-chip memory system, schedule data transfers, and balance the off-chip memory system, on-chip interconnection and on-chip memory system.

In this paper, we try to give a guidance for how to offer a well-designed on-chip memory system through task scheduling. After our experiments, we find some relationships among the capacity of memory system, the number of memory banks and the frequency of interconnection.

Following the above discussion, the remainder of this paper is organized as flows. We start by introducing related work in next Section II. Then we describe task scheduling in Section III and memory system in Section IV. In Section V, we show the experiments and results. Finally Section VI is a conclusion.

II. RELATED WORK

When designing a microprocessor, the most important step is to decide appropriate design configurations satisfying different performance, power and reliability constraints, which is called as Design Space Exploration (DSE) [5]. Finding a desirable design in a large amount of candidates is difficult, as each design holds the performance, area and energy consumption trade-off. Therefore, the DSE problem is introduced to search for architecture candidates within a short time [6]. Most DSE methods based SystemC which is a library based on the object oriented programming language C++. It offers system modeling at different abstraction levels which can influence the simulation speed.

Some research use flexible model in simulation to accelerate the estimation for various architectures. [7] propose a bus architecture independent non-structured simulation model for performance evaluation. The model detects the start time, the quantity of data and the destination information of each transaction. Through the model based on SystemC, their method can speed up 28x than FPGA.

[8] propose a performance evaluation method using a communication analysis graph (CAG). CAG describes data processing and data transfers as nodes and dependencies between the data processing and the data transfer as edges. It is analyzed for performance estimation. These models are flexible for timing analysis of various bus architectures because they are constructed once for a functional block set.

[6] propose a research of system design method based on system-level profiling. They use a system level model to explore the parameters set search tree and evaluates the design quality of each architecture candidate. Their works make two contributions. The one is the execution time estimation based on pipeline, burst and split transaction of high-performance bus protocols. The other is the communication architecture exploration of standard bus specification, resulting in a realistic architecture rather than an abstract bus model of the preceding research. In experiments, IP database, the number of ports, execution time, execution sequence, bus specification, bus width and frequency are input parameter. The output are the architecture level model. The task object is compressing a 512 x 512 pixels image. The results can give a design guidance to bus specification for pipeline, burst and split behavior.

[9] propose a DSE method based on Regression Random Forests (RRF) which can build accurate and reliable prediction models. Their method can significantly improve the prediction accuracy and accelerate the procedure. In addition, due to the comprehensible tree model, RRF could guide the SoC design by ranking the parameters' importance. The experimental results show that RRF can reduce relative errors by about 60%, provide with high accurate predictions in less time and offer useful guide information due to its feature-ranking attribute.

[5] propose the COMT approach which can exploit unlabeled design configurations to improve the models. In addition to an improved predictive accuracy, COMT is able to guide the design of microprocessors, owing to the use of comprehensible model trees. Empirical study demonstrates that COMT significantly outperforms state-of-the-art DSE technique through reducing mean squared error by 30% to 84%, and thus, promising architectures can be attained more efficiently. Furthermore, the simulation speed can also be improved by COMT.

In [10], a modular framework is proposed and modeled for the exploration and evaluation of SoC system based on SystemC. In addition, the proposed methods consider optimizing the power as one of the design constraints. Various design variations are implemented and evaluated using standard task graphs such as Directed Acyclic Graph (DAG). And performance evaluation metrics are evaluated and discussed for various design scenarios.

[11] propose an architecture performance estimator of which the input parameters are matrix's size, the number of cores, pipeline's size, bandwidth and memory capacity. Through the algorithm mapping module and the distribution scheme of matrix in different memory system, the estimator can balance the input parameters to make cores near the peak performance.

[12] propose an open source framework---Multicube Explorer which can optimize the memory system in multi-core architecture by various situations. The optimization algorithm adopts traditional heuristic algorithms and multi-objective optimization algorithms and it can evaluate the memory capacity, power consumption and throughput bandwidth.

Most of the existing methods focus on the acceleration and a few focus on the power saving. They all do not map and schedule the applications. Thus the results may not find the real bottleneck of multi-core processors. In other words, some redundancy may exist in their multi-core architecture.

III. TASK SCHEDULING

The task scheduling algorithm in this work adopts Homogeneous Earliest-Finish-Time (HoEFT) algorithm which is based on list scheduling and proposed in our previous work. In HoEFT, the cores are homogeneous and their execution time for each sub-task are known before scheduling. The capacity of each core's memory and shared memory, and the number of shared memory's bank are set by designers. The bus in HoEFT is assumed to be crossbar, and the frequency and data width can also be set to any size. The object of HoEFT is to minimize the total execution time, and we do not care power. Compared with traditional task scheduling algorithms, HoEFT introduce two limiting conditions: traffic-aware and memory aware. Through these two conditions, HoEFT can simulate the real system accurately. In addition, we introduce indexes to determine which data can be overwritten when the memory is full.

HoEFT three phases: task prioritizing phase, core selection phase and update phase. Task prioritizing phase is completed according to the rank value of each tasks. The rank value is calculated by the execution time and communication time. In core selection phase, the first step is to find the earliest execution time through insertion-based algorithm. Then the data needed by the current should be transferred temporarily. At the same time, the ports and memory capacity should also be occupied temporarily. Through that way, HoEFT find the final execution time of the current task. After the above phase, HoEFT will update every parameter so that the system can continue to execute.

In previous work, we built an eight-core system with some fixed parameters such as memory banks, capacity and bus frequency. Under previous constraints, every core can get near-peak performance.

IV. MEMORY SYSTEM

In this work, we assume that memory system is consist of three parts: off-chip memory, shared memory and cores' memory. Off-chip memory is DRAM which will be described in Section V. Shared memory and cores' memory are SRAM which can be read or written directly. The memory model is Scratch Pad Memory (SPM) of which the address is fully visible for the programmer. Through this layering memory, the multi-core architecture can pre-fetch data at each layer. Then if the bus can provide enough bandwidth, the cores can get the near-peak performance (as the data can be always ready before the usage of cores).

However, the memory capacity and the bus frequency cannot be infinite great. Although oversize capacity can bring improvement in performance, it would lead to a large area and decrement of yield. The delay of the SRAM unit would increase which can lead to decrement in SRAM frequency. In addition, oversize area makes the timing closure of bus difficult to be completed in back-end design and power consumption to be higher than normal. Conversely, though the small capacity can lead to low power and high frequency, undersized capacity would result in substandard performance.

SRAM is usually generated by memory compiler. However, the ports of these SRAM are at most 2. In order to provide enough bandwidth for multi-core architecture, the memory system should contain many SRAM banks or memory banks. Although more memory banks can bring higher bandwidth for system, too many banks would lead too much pressure to the bus which can also make the timing closure of bus difficult to be completed in back-end design. Thus we should find the relationship among the memory capacity, the number of memory banks and bus bandwidth.

V. EXPERIMENTS AND RESULTS

In order to give a guidance for the design of memory system, we use the HoEFT algorithm to find the shortest execution time for a given multi-core system. In this work, our target task is Double-precision General Matrix Multiplication (DGEMM) which is an important part in Linpack. The multi-core architecture consists of eight cores, shared memory, off-chip memory and a crossbar.

The core model is based on our existing APE core which stands for Algebraic Processing Engine [13]. The core has been improved in 28 nm process which contains four 512-bit Floating-point Multiply-Add (FMA) units. At the back-end design, the core can run at 1.4 GHz. For DGEMM, the peak performance of each core is 89.6 Gflops. We assume that each core contains m_1 MB memory. Since the capacity of each core should not too large, we assume m_1 should satisfy the inequality (1).

$$0.5 <= m_1 <= 2 \quad (1)$$

We assume that the shared memory consists of n memory banks. Each of the memory bank is single port which can be modeled in HoEFT. The total capacity of shared memory is assumed to m_2 MB. Considering the on-chip memory of chips which has been successfully taped-out, we assume that n should satisfy the inequality (2) and m_2 should satisfy the inequality (3).

$$8 <= n <= 32 \quad (2)$$

$$8 <= m_2 <= 32 \quad (3)$$

We assume that the off-chip memory is DDR4 as our laboratory has bought the IP from Synopsys. We assume that the SDRAM runs at 2800 MHz with 64-bit data width and we don't care about the capacity of SDRAM. We assume there are two DDR4 channels in the multi-core architecture, and only one channel can provide data transaction and the other is prepared for the high-speed IO (such PCIe). Thus the off-chip memory system can provide 89.6 Gbps bandwidth (we assume that the utilization of DDR4 is 0.5).

The interconnection in this multi-core architecture is a crossbar. We assume that each port of the crossbar can handle read and write at the same time like Advanced eXtensible

Interface (AXI). We assume that the data width of each port is 128 bits and the frequency of the crossbar is f GHz. In consideration of the frequency of cores, we assume that f should satisfy the inequality (4).

$$0.2 <= f <= 1.4 \quad (4)$$

Under different parameters (m_1, m_2, n, f) combinations, we use HoEFT to schedule DGEMM. We consider that when the parameters can make the core's wait time less than 5% of the total execution time, they can be selected as one solution. After a lot of experiments, we find all the solutions under our constraints and show them in Figure 1 and Figure 2. We can see that if these parameters are large enough, cores can run fully without wait time. We can also choose the appropriate designing parameters for this multi-core processor from the solutions. In addition, we find that for the optimal solutions, m_1, n and f are inversely proportional to each other, and m_2 is inversely proportional to f.

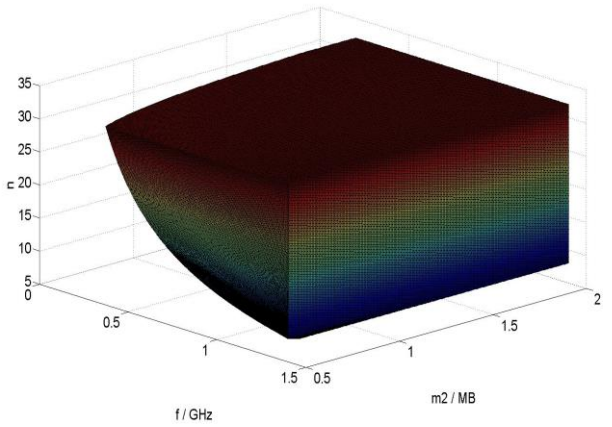

Figure 1. The solution set for m_1, n and f.

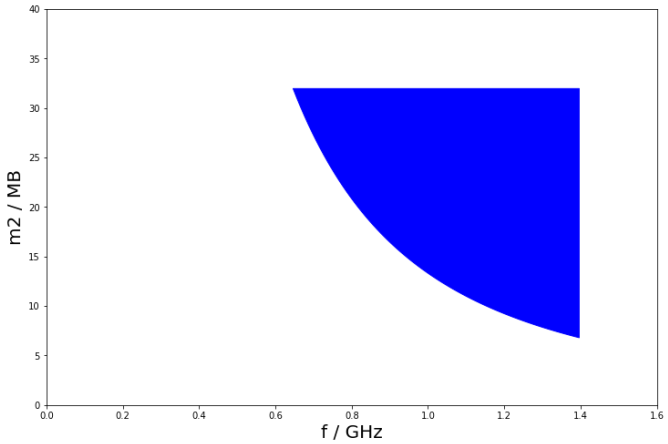

Figure 2. The solution set for m_2 and f.

VI. CONCLUSION

In this paper, we propose a brand new DSE method for on-chip memory system based on task scheduling. Through this method, we find some relationships among the capacity of memory system, the number of memory banks and the frequency of bus when the target application is DGEMM. In addition, the solution set of parameters can instruct the designers to provide an appropriate multi-core architecture. Like our method, compilers of computers can also give some guide for the architecture design, but traditional compilers do not check memory system and multi-core communication, and they only know the computing units, registers and programme.

In this paper, we only complete the DSE for memory system under DGEMM using this method. In the future work, we will explore more parameters such as the number of cores and the number of DDR channels. In addition, we will also add more target applications to give a more comprehensive design guidance.

REFERENCES

[1] Present I. Cramming more components onto integrated circuits [J]. Readings in computer architecture, 2000, 56.

[2] Zheng J, Wu X. The Compiler-Based Heterogeneous Multi-core Structure [C]// International Conference on Multimedia Communications. IEEE Computer Society, 2010:53-56.

[3] Hardee K C, Chapman D B, Pineda J. Dynamic random access memory: U.S. Patent 5,077,693 [P]. 1991-12-31.

[4] Yamaguchi S, Ichinohe E, Katsura J. Static random access memory: U.S. Patent 4,712,194 [P]. 1987-12-8.

[5] Chen T, Chen Y, Guo Q, et al. Effective and efficient microprocessor design space exploration using unlabeled design configurations [J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2013, 5(1): 20.

[6] Sombatsiri S, Kobashi K, Sakanushi K, et al. An AMBA hierarchical shared bus architecture design space exploration method considering pipeline, burst and split transaction [C]// Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on. IEEE, 2013: 1-6.

[7] Kuehnle M, Brito A, Roth C, et al. An approach for power and performance evaluation of Reconfigurable SoC at mixed abstraction levels [C]// Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on. IEEE, 2011: 1-8.

[8] Lahiri K, Raghunathan A, Dey S. System-level performance analysis for designing on-chip communication architectures [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2001, 20(6): 768-783.

[9] Lin C, Du X, Jiang X, et al. An efficient and effective performance estimation method for DSE [C]// VLSI Design, Automation and Test (VLSI-DAT), 2016 International Symposium on. IEEE, 2016: 1-4.

[10] Tafesse B, Muthukumar V. Framework for simulation of heterogeneous MpSoC for design space exploration [J]. VLSI Design, 2013, 2013: 11.

[11] Kai Z. Design Tradeoffs of High Performance DSPs for General-Purpose HPC[J]. Chinese Journal of Computers, 2013, 36(4):790-798.

[12] Zaccaria V, Palermo G, Castro F, et al. Multicube Explorer: An Open Source Framework for Design Space Exploration of Chip Multi-Processors[C]// International Conference on Architecture of Computing Systems. VDE, 2011:1-7.

[13] Wang D, Du X, Yin L, et al. MaPU: A novel mathematical computing architecture [C]// High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on. IEEE, 2016: 457-468.