Microarchitecture and Performance Analysis of Godson-2 SMT Processor

Zusong Li, Xianchao Xu, Weiwu Hu and Zhimin Tang
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, 100080
{lisoon, xuxianchao, hww, tang}@ict.ac.cn

***Abstract*—This paper introduces the microarchitecture and logical implementation of SMT (Simultaneous Multithreading) improvement of Godson-2 processor which is a 64-bit, four-issue, out-of-order execution high performance processor. The condition for implementing correct memory consistency model in Godson-2 SMT processor is studied and a new register-level sharing and synchronization scheme is proposed. Godson-2 SMT processor has been implemented at the RTL level and simulated with the VstationPro of Mentor Graphics. The Linux operating system is ported to run in Godson-2 SMT processor and application programs such as SPEC CPU2000 benchmark suite are used to evaluate performance. Experimental results indicate that the performance of Godson-2 SMT processor is improved significantly by fully exploiting thread-level parallelism and optimized utilization of functional units. The average speedup is 31.3% with 18.8% area overhead.**

***Index Terms*—Godson-2, Simultaneous multithreading, Microarchitecture, Memory consistency model, Register sharing**

I. Introduction

THE rapid development of semiconductor technology drives the improvement of processor architectures. As a single chip contains over one billion transistors, the exploitation of thread-level parallelism becomes the trend of high performance microprocessor design. Several multithreading processors are implemented by industry in previous years [1]-[3].

Simultaneous multithreading (SMT) [4]-[6] inherits the ability to issue multiple instructions each cycle from superscalar, while utilizing independent instructions of multiple threads to find more instructions executing in parallel. It uses its resources more efficiently and thus achieves better performance than its conventional superscalar counterpart by adding minimal hardware complexity and chip area. SMT exploits thread-level parallelism fully in exchange of slight hardware cost.

Godson-2 processor [7] is a four-issue superscalar and nine-stage superpipelining microprocessor. Exploitation of instruction-level parallelism (ILP) is achieved by the adoption of aggressive branch prediction, register renaming, out-of-order execution, non-blocking cache, and load speculation techniques. This paper introduces the SMT improvement of Godson-2 processor. It presents microarchitecture of the Godson-2 SMT processor, studies condition for implementing correct memory consistency model, and proposes a new register-level sharing and synchronization mechanism.

The following of the paper is organized as follows. Section 2 elaborates the microarchitecture of Godson-2 SMT processor. Section 3 describes the memory consistency model adopted. Section 4 describes the mechanism for register sharing and synchronization. Section 5 describes the experimental methodology of performance evaluation. Section 6 briefly presents the physical design and evaluates the area of the chip. Section 7 evaluates the performance. Finally, conclusions and directions for future work are given in section 8.

Manuscript received May 4, 2006. This work was supported by the National Natural Foundation of China for Distinguished Young Scholars under Grant No.60325205; the National High-Tech Research and Development Plan of China under Grant Nos. 2002AA110010, 2005AA110010 and 2005AA119020; National Basic Research Program of China under Grant No. 2005CB321601.

II. Architecture of Godson-2 SMT processor

Godson-2 SMT processor supports the simultaneous execution of two threads and the following two running models, Superscalar model and SMT model.

Like Godson-2 superscalar processor, the superscalar model executes one thread using all hardware resources, including various queues, pipeline path, physical register file, functional units, caches, and so on.

SMT model executes two threads generally coming from different programs. Each thread owns its individual program counter (PC), logical registers, and control registers. Two threads share other hardware resources, such as various queues, pipeline path, functional units, cache, and so on. The share scheme consists of proportional sharing, full sharing, and time-multiplexing sharing. The proportional sharing means that each thread only uses half of the resources, for example, using half number of the reorder queue entries. The full sharing scheme means each thread can use the most of the resources, but cannot use all the resources, for example, the use of fixed-point reservation station is in this class. The time-multiplexing sharing means that each thread uses the resources by turns, for example, using functional units and pipeline path.

Godson-2 SMT microarchitecture is similar to conventional simultaneous multithreading processors. However, there are a number of tradeoffs among the performance, the difficulty of physical design and the cost of hardware resources in considering the implementation of Godson-2 SMT processor. For example, to reduce the chip area, to decrease the latency,

and to ease physical design, Godson-2 SMT processor uses single-port RAM to implement instruction cache. The area of RAM is in the direct ratio to the number of ports. Table I shows that the area of dual-port RAM is 2.8 times that of single-port RAM. The latency of dual-port RAM is also longer than that of single-port RAM. Godson-2 SMT processor only fetches instructions from one thread each cycle in round robin policy. However, conventional simultaneous multithreading processors use multiple-port RAM to implement instruction cache. They fetch instructions from two different threads in each cycle. In addition, the physical register file in conventional simultaneous multithreading processors is difficult for physical implementation. Furthermore, the larger register file requires longer access time. To avoid increasing the processor cycle time, Godson-2 SMT processor allocates individual register file to each thread. Though resource is wasted in a certain extent, high clock frequency is easy to be realized.

TABLE I
COMPARISON BETWEEN SINGLE-PORT AND DUAL-PORT RAM OF 0.18UM CMOS

	Single-port RAM	Dual-port RAM
area (um2)	216258	611910
latency (ns)	1.384	1.547

Fig. 1 shows the microarchitecture of Godson-2 SMT processor. It supports two threads and contains the contexts of two threads. Some hardware resources added to Godson-2 SMT processor include PC, fixed-point register rename, floating-point register rename, fixed-point reservation station, floating-point reservation station, control registers, branch queue, and so on. Other resources are shared between two threads due to different sharing policies. Each thread uses half of the resources related to context, including instruction TLB (Translation Lookaside Buffer), reorder queue, CP0 queue (Memory Access Queue). The partition sharing scheme avoids one thread interfering the other. Instruction cache (I-Cache), data cache (D-Cache), branch target buffer (BTB), branch pattern history table (PHT), data TLB and functional units are shared completely. Both threads share fixed-point reservation station and floating-point reservation station. When an instruction enters the reservation station, an empty entry is randomly allocated for it. However, the number of entries each thread can use is restricted that no less than four entries will be left to allocate to the other thread. When a thread is blocked during its execution by remote memory accesses, cache misses, or synchronization needs, this sharing policy ensures the instructions of the other thread can be issued.

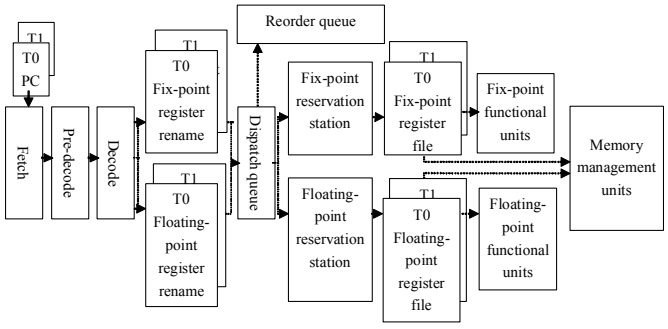

Fig. 1: Microarchitecture of Godson-2 SMT processor

Godson-2 SMT processor fetches new instructions from the selected thread and sends them to IR (Instruction Register) in each cycle. The instructions selected from one thread are decoded into internal instruction format of Godson-2 and are sent to the register renaming module. The renamed instructions are sent to reservation stations and queues. The instructions with all required operands ready are selected from the fixed or floating-point reservation station for each functional unit. Then the instructions are executed and the results are written back. Finally, reorder queue graduates the instructions of each thread in program order individually.

III. MEMORY CONSISTENCY MODEL

Memory consistency model influences the performance and the complexity or cost of the design. This section is dedicated to implementation issues of Godson-2 SMT processor related to various memory consistency models including weak ordering (WO) [8], sequential consistency (SC) [9], and processor consistency (PC) [10].

The primary data cache is shared across the two threads of Godson-2 SMT processor. Once data is stored to data cache, memory consistency need not be concerned. The operation of memory instructions in CP0 queue must be considered to satisfy memory consistency model. Godson-2 superscalar processor has already implemented SYNC instruction. Simply using synchronization operations to coordinate memory operations on different threads and maintaining program order, weak order consistency model is able to be implemented. Godson-2 SMT processor ensures that memory access instructions graduate in the order specified by the program. Load instructions execute before graduating and store instructions write their values in sequential order after graduating by the design of memory access queue. In this way, constraints on event orderings imposed by processor consistency are satisfied. Therefore, Godson-2 SMT processor also supports processor consistency model.

Clearly, Godson-2 SMT processor does not satisfy sequential consistency model. To guarantee its compatibility with software, memory consistency model in Godson-2 SMT processor should be sequential consistent and ensures that the execution result of multiple threads is the same as if the operations are executed in uniprocessor. Otherwise, the execution result may be incorrect. Take the program segment in Fig. 2 as an example. After this program segment is executed, the correct combinations of values for register R_1 and R_2 are (0, 1), (1, 0) or (1, 1). The result $R_1 = R_2 = 0$ is incorrect.

Thread T_0	Thread T_1
L_{11} : store a , 1	L_{21} : store b , 1
L_{12} : load R_1 , b	L_{22} : load R_2 , a

Fig. 2: Example program segments PRG (initially $R_1 = R_2 = a = b = 0$)

A store is write-ready when the value to store is valid and it has been committed (that is, cannot be cancelled) in Godson-2 superscalar processor. Only the write-ready store instruction can write data cache. If data cache miss occurs, the store is in CP0 queue temporally, and it is impossible for the other thread which can commit the load instructions following this store to access the value of the store. This condition is clearly not sufficient for satisfying sequential consistency. If both threads meet the above condition, they read the value $R_1=R_2=0$. If the

value written by store becomes accessible by both threads concurrently in order of graduate, this error possibility can be precluded.

When a load enters the CP0 queue, it checks all the older stores for possible bypass for each byte it needs as Godson-2 superscalar processor. Furthermore, it checks whether there is dependent stores in the graduated stores queue. When a store enters the CP0 queue, it checks all the younger loads in front of another younger store to the same byte in the queue to decide whether to forward value to them. An exception arises and the loads are cancelled, if the loads dependent to the store have written back. The primary difficulty of this mechanism is in hardware overhead and possible correctness problems. Firstly, each thread commits multiple instructions in one cycle. If multiple stores commit in one cycle, they all need to check the CP0 queue which incurs a significant overhead. Secondly, if the dependent store and load from two threads commit at the same time, the exception will not arise and error occurs. For example, consider the program segment in Fig. 2. If four instructions commit in one cycle, error arises.

The above problem can be solved efficiently by implementing additional function of CP0 queue for Godson-2 SMT processor as follows. When a load enters the CP0 queue, it checks the stores of the other thread for possible dependency. An exception arises and load is cancelled if dependency exists. When a store enters the CP0 queue, it checks all the loads to the same byte in the CP0 queue of the other thread. An exception arises and store is cancelled if dependency exists. In this way, the implementation guarantees that the dependent load and store instructions from two threads will not be processed at the same time, and ensures that the value written by one thread becomes accessible by the other thread. As for store and store dependency, in-order graduate ensures the correctness. Load and load dependency does not need to be detected because it may not lead to errors.

In summary, WAW dependency is resolved by in-order operating of stores and forwarding the value of store to the following loads in the graduate store queue. Furthermore, RAW dependency is resolved by checking RAW dependency between threads and raising exception in the condition of dependency existence. Hence, Godson-2 SMT processor supports sequential consistency model by the technique discussed above.

IV. Register Sharing and Synchronization

Godson-2 SMT processor implements full/empty synchronization to pass messages between threads at register level. Each register has an associated full/empty bit. Each register can be read and written by synchronized read and synchronized write instruction. Synchronized read and write instruction can only be executed when condition is satisfied. Taking into account complex circumstances, for example in the mispredicted branch canceling case, synchronized write instruction of one thread is not allowed to write the other thread's register.

Full/empty scheme has an issue that should be considered. A synchronized read instruction is in register renaming stage after decoded, and the register it reads is empty. If the instruction waits for the register it reads to be set to full in register renaming stage, it will block the pipeline and result in deadlock. If synchronized read instruction enters reservation station first, its physical register number is not the number of synchronized write instruction which will write this physical register in the future. The synchronized read instruction needs to be renamed again. Thus the logical register number of synchronized read instruction must be known, and new rename result must inform reservation station to modify the physical register number of the synchronized read instruction. The above disposition has a large impact on complexity of design.

To implement full/empty scheme, the problem that the register for synchronized read/write instruction the unready should be solved. Our effective solution is blocking synchronized read/write instructions in instruction buffer in decode stage. This scheme not only avoids blocking the whole pipeline, but also renames the register to get correct physical register number after the register is ready. In pre-decode stage, synchronized read/write instruction is decoded. At the same time, destination register number is decoded. Only the instruction whose register is ready can be chosen to enter decode stage. If synchronized read instruction is used before corresponding synchronized write instruction by programmer's error, the processor will wait endless in decode stage. When interrupt occurs, the instruction whose register is unready will be selected and signed exception to avoid deadlock. To avoid mispredicted branch canceling, synchronized read/write instructions do not set full/empty bit until commit stage. Full/empty synchronization not only can pass messages at register level, but also can synchronize between two threads.

V. Experimental Methodology

Godson-2 SMT processor is improved step by step. The base processor models of Godson-2 SMT processor are summarized in Table II. The above design of the Godson-2 SMT processor has been implemented at RTL level based on verilog hardware description language. Godson-2 SMT processor has been logically synthesized to evaluate the chip area.

TABLE II
Processor models

Fetch width	4	Cp0queue	32
Decode width	4	DTLB	64
ALU	2	ITLB	16
FP Unit	2	BHT	4K
MMU	1	Data Cache	64KB
Roqueue	64	Instruction Cache	64KB
Fixqueue	16	On-chip Secondary Cache	No
Ftqueue	16		

The performance of Godson-2 SMT processor is evaluated by RTL simulation. To build the simulation systems, the motherboard is RTL implemented based on verilog hardware description language. The motherboard environments implement the peripheral components, including SYSAD interface, 1M off-chip secondary cache, 1M ROM, 32M RAM (memory) and serial interface (the characters are output to serial interface by processor and are printed to screen by serial interface). The latency of secondary cache access is 2 cycles. The latency of memory access is 16 cycles. VStationPro

simulation accelerator developed by Mentor Graphics Company was used to accelerate the running of RTL codes.

Because the performance of superscalar model of Godson-2 SMT processor is the same as Godson-2 superscalar processor, our performance evaluation is based on the executing results of superscalar model and SMT model of Godson-2 SMT processor. Linux2.4.20 operating system is modified to work on Godson-2 SMT processor, and application programs are executed in it to evaluate performance. Because memory resource of VStationPro is limited, only the programs whose memory resource requirement is little can be used in this study. We use the following three kinds of programs to evaluate performance. The first kind is a program (Inst Depend Program) that all instructions are dependent. This program is executed twice in superscalar model. Two threads execute this program at the same time in SMT model. The second kind of program is the multithreading chat program. The third kinds of program are eon, twolf, art, crafty from SPEC CPU2000 benchmark suite. We choose two programs running together in superscalar model and SMT model. Total number of instructions and total number of cycles are statistically collected for each model, and IPC (Instruction per Cycle) is calculated to evaluate the performance.

VI. Chip Area Evaluation

Chip area is the major factor that influences chip cost, and the manufacturing cost of Godson-2 SMT processor is best measured by its chip area. The design of the Godson-2 SMT processor is physically synthesized based on 0.18um CMOS process to evaluate the chip area. Fig. 3 shows chip area comparison of Godson-2 SMT processor and Godson-2 superscalar processor.

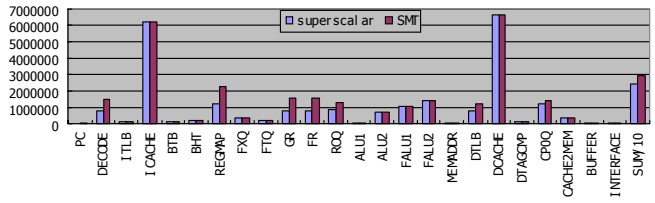

Fig. 3: Chip area comparison

As is shown in Fig. 3, the majority of the SMT area overhead comes from the following modules: PC, DECODE, REGMAP (Register Mapping), GR (Fixed-point Physical Register File) and FR (Float-point Physical Register File), because the logic of these modules are doubled due to the addition of SMT. For example, doubling PC, instruction buffer, register renaming and register file takes more than twice the area of corresponding modules. Adding SMT will double branch queue in ROQ module, double control register set in DTLB, and add dependency checking logic to resolve conflicting accesses between threads in CP0. Hence, the area of ROQ (Reorder Queue), DTLB (Data TLB) and CP0Q (Memory Access Queue) also grows. The area of cache and functional units of Godson-2 SMT processor is equal to Godson-2 superscalar processor, because no additional logic is added to these modules.

Comparing to Godson-2 superscalar processor, Godson-2 SMT processor only adds chip area in 18.8%. Relative percentage of the area increased by adding SMT in Godson-2 SMT processor is bigger than Hyperthreading Xeon of Intel. The reason is that Xeon processor has on-chip secondary cache, and its front-end decode module is complex. The area of on-chip secondary cache and decode sharing between two threads occupy the largest percentage of entire chip area. Adding SMT will not multiply the on-chip secondary cache and decode area overhead. The secondary cache always occupies more than half of entire chip area. Therefore, relative percentage of the increased area of Xeon is smaller. Suppose Godson-2 processor implements on-chip secondary cache, the added area will be less than 10 percent.

VII. Performance Evaluation

A. Single-Program Benchmarks Performance Analysis

We execute single-program benchmarks under simultaneous multithreading model and superscalar model to evaluate the performance of single-program workloads. Fig. 4 presents the IPC of single program benchmarks in SMT and superscalar. Experimental results show IPC of superscalar is larger than SMT. The main reason is that each thread shares reorder queue and CP0 queue in partition. Half the entries of these queues are idle when running single program workload in SMT. Of course, most of the resources are full sharing and can be used by one thread in SMT under single program workload. Hence, the performance of SMT is close to superscalar under single program benchmarks. In fact, Godson-2 SMT processor supports real time switch between SMT model and superscalar model. Operating system can detect single program workload and switch from SMT model to superscalar model adaptively to attain better performance.

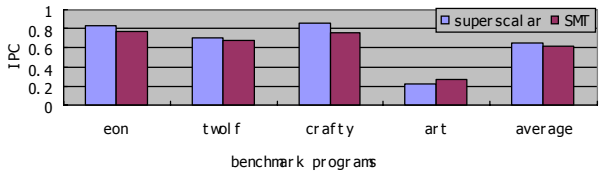

Fig. 4: Performance of single program benchmarks

B. Perfect Performance Analysis

Godson-2 SMT processor supports two threads simultaneously. Therefore, the perfect IPC of Godson-2 SMT processor is twice the IPC of Godson-2 superscalar processor. Table III shows the IPC of instruction dependent program in superscalar and SMT. Experimental results in Table III show that Godson-2 SMT processor makes full use of functional units. When one thread waits for the calculated results, the other thread can use functional units to complete its task. Thus the running time of two programs approximates to the running time of one program. Experimental results show that Godson-2 SMT processor obtains the expected perfect performance.

TABLE III
EXPERIMENTAL RESULTS OF INSTRUCTION DEPENDENT PROGRAM

IPC of superscalar	IPC of SMT	performance speedup
0.799873913	1.598539529	1.998

C. Per-pipeline Stage Performance Analysis

We execute the combination of two programs from eon, twolf, crafty and art program of SPEC CPU2000 benchmarks in Godson-2 SMT processor to evaluate the performance of pipeline stages. The experimental results are shown in Fig. 5. We can see from Fig. 5 that the IPC value of pipeline stage is less than the IPC value of its previous stage except fetch stage and decode stage. Because there is an instruction buffer between fetch stage and decode stage, Godson-2 SMT processor can still decode the blocked instructions from instruction buffer, when fewer instructions are fetched from instruction cache. Godson-2 SMT processor uses single-port RAM to implement instruction cache, which avoids the area overhead of dual-port RAM. Instruction buffer balances the negative effects of fetch stage in most cases, though only one thread can fetch instructions from instruction cache per cycle. Hence, the fetch policy of Godson-2 processor achieves sufficient performance with low cost.

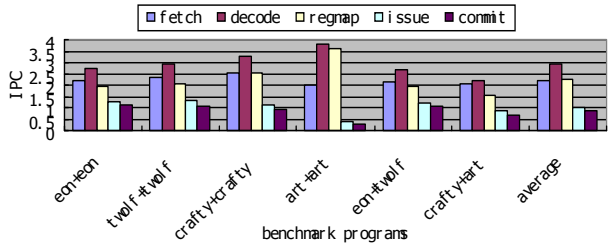

Fig. 5: Pipeline stages performance analysis of Godson-2 SMT processor

The implementation policy of fetch stage and decode stage in Godson-2 SMT processor is different from the policy in conventional SMT processors. We will analyze decode stage performance in the following figure. Fig. 6 shows the IPC distribution of decode stage. In Fig. 6, lossinst-1 denotes the case that the amount of instructions decoded by one thread is less than 4 and instruction buffer of the other thread has instructions. Lossinst-2 denotes the case that the amount of instructions decoded by one thread is less than 4 and the instruction buffer of the other thread is empty. Noinst denotes the case that there is no instruction in both threads. Considering IPC degradation in above cases, only lossinst-1 can be avoided by fetching instructions from both threads. However, the performance loss due to lossinst-1 is only 0.13, which occupies low percentage of total performance loss. On the other hand, the cost and complexity of fetching instruction from both threads is high. Hence, the decode policy of Godson-2 processor achieves high cost-performance. The majority losses of IPC in decode stage due to the case that the instructions in instruction buffer are not sufficient. Improving the ability of instruction fetching is a way of achieving high performance.

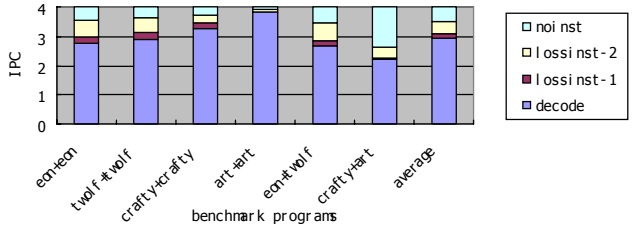

Fig. 6: Decode stage performance analysis of Godson-2 SMT processor

D. Multi-Program Benchmarks Performance Analysis

We also execute multi-program benchmarks in SMT and superscalar to evaluate the performance under multi-program workloads. The multi-program benchmarks include multithreading application program chat and the combination of SPEC CPU2000. Fig. 7, 8, 9, 10 clearly show branch misprediction rate, instruction cache miss rate, data cache miss rate and data TLB miss rate in SMT and superscalar. These resources are shared between two threads, and one thread interferes with the other thread. Hence, various miss rates increase a little in SMT.

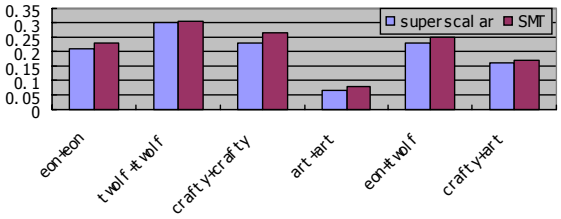

Fig. 7: Branch misprediction rate of multi-program benchmarks

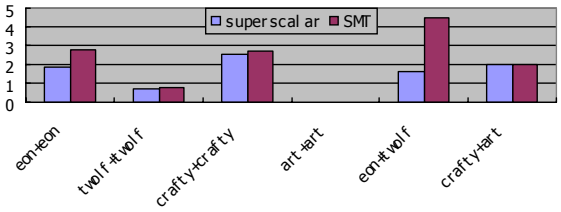

Fig. 8: Instruction cache miss rate of multi-program benchmarks

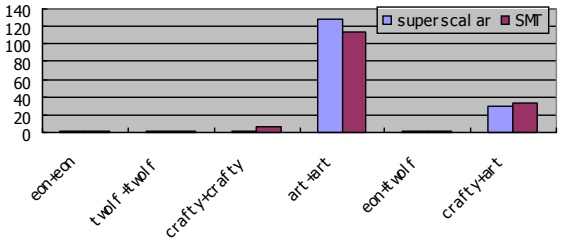

Fig. 9: Data cache miss rate of multi-program benchmarks

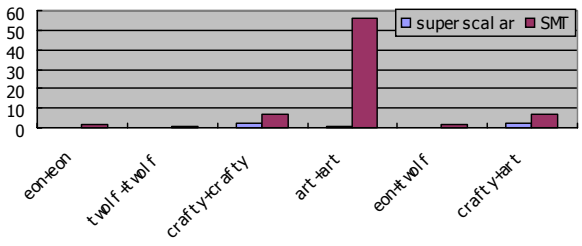

Fig. 10: Data TLB miss rate of multi-program benchmarks

The resources of various queues are also shared between two threads. The available queue resources of each thread in SMT are less than the resources in superscalar, and the full rates of queues are larger. However, when one thread blocks, the SMT processor can fetch instructions from the other thread. Thus the block times of SMT are less than the block times of superscalar. Fig. 11 shows the pipeline block times of multi-program benchmarks in SMT and superscalar. T0 and T1 in Fig. 11 are used to denote the queue full times of thread 0 and thread 1. We can see from Fig. 11 that queue full times of each thread in SMT are more than that in superscalar. But queue full times of both threads which indicate the block times of multithreading

processor in SMT are less than that in superscalar. Furthermore, the results show that Godson-2 SMT processor makes full use of the resources by executing the instructions of one thread, when the other thread is blocked.

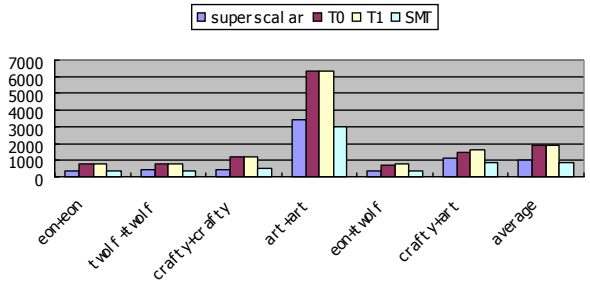

Fig. 11: Pipeline block times of multi-program benchmarks

Fig. 12 presents the IPC of multi-program benchmarks in SMT and superscalar. The results show Godson-2 SMT processor improves performance efficiently. The speedup on average is 31.1%. The performance improvement of twolf+twolf is up to 54.4%. Though branch misprediction rate, cache miss rate and TLB miss rate of SMT processor is a little higher than those of superscalar processor as a result of resources competition, Godson-2 SMT processor improves performance significantly by full exploitation of thread-level parallelism. Godson-2 SMT processor utilizes fully the independency characteristic between multithreads, therefore exploits more instructions executing in parallel in the same instruction window. High utilization rate of functional units increases the performance of Godson-2 processor effectively.

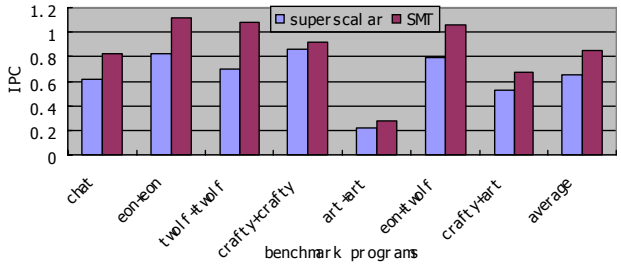

Fig. 12: Performance of multi-program benchmarks

VIII. Conclusions and Future Work

This paper elaborates the microarchitecture of the Godson-2 SMT processor. Memory consistency model and register sharing scheme which influence the performance and the complexity of the design are described in detail. Some application programs of SPEC CPU2000 are used to evaluate the performance. It has been shown that Godson-2 SMT processor improves the performance of Godson-2 superscalar processor significantly by fully exploiting thread-level parallelism, high utilization rate of functional units and fast register sharing and synchronization. The average speedup is 31.1%. Our future work includes improving performance of Godson-2 SMT processor and exploiting further multithreading parallelism through putting multiple Godson-2 SMT processors on the same chip.

References

[1] Deborah T.Marr, Frank Binns, David L. Hill, Glenn Hinton, David A. Koufaty, J. Alan Miller, and Michael Upton. Hyper-Threading Technology Architecture and Microarchitecture. Intel Technology Journal Q1, 2002, pp. 4-15

[2] Ron Kalla, Balaram Sinbaroy, and Joel M.Tendler. IBM Power5 Chip: A Dual-Core Multithreaded Processor. IEEE Micro, Mar./Apr. 2004, pp. 40-47

[3] Poonacha Kongetira, Kathirgamar Aingaran, Kunle Olukotun. Niagara: A 32-Way Multithreaded Sparc Processor. IEEE Micro, Mar./Apr. 2005, Vol. 25, No. 2, pp. 21-29

[4] Tullsen D.M., Eggers S.J., Levy H.M. Simultaneous Multithreading: Maximizing On-Chip Parallelism. Proceedings of 22nd Annual International Symposium on Computer Architecture, Santa Margherita Ligure, Italy, Jun. 1995, pp. 392-403

[5] Tullsen D.M., Eggers S.J., Emer J.S., Levy H.M., Lo J.L., and Stamm R.L. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor. Proceedings of 23nd Annual International Symposium on Computer Architecture(Philadelphia, PA) , 1996, pp. 191-202

[6] Eggers S.J., Emer J.S., Levy H.M., Lo J.L., Stamm R.L. and Tullsen D.M. Simultaneous Multithreading: A Platform for next-generation processors. IEEE Micro, 1997, Vol. 17, No. 5, pp. 12-19

[7] Weiwu Hu, Fuxin Zhang, Zusong Li. Microarchitecture of the Godson-2 processor. Journal of Computer Science and Technology, Mar. 2005, Vol.20, No.2, pp. 243-249

[8] Dubios M, Scheurich C, Briggs F. Memory Access Buffering In Multiprocessors. Proceedings of the 13th International Symposium on Computer Architecture, 1986

[9] Lamport L. How to Make a Multiprocessor Computer That Correctly Executes Multiprocessor Programs. IEEE Transactions on Computers, 1979, Vol. C-28, No. 9

[10] Goodman J. Cache Consistency and Sequential Consistency. Technical Report No. 61. SCI committee, 1989

Zusong Li received his B.S. degree from Tsinghua University in 1996 and his Ph.D. degree from the Institute of Computing Technology, the Chinese Academy of Sciences in 2006, both in Computer Science. He is one of the three architectural designers of Godson-2 processor, and his interest focuses on high performance computer architecture, verification and VLSI design.