In Journal of Information, Vol.9, No.1, January, 2006

Partitioned Register File Designs for Clustered Architectures

Yukinori Sato, Ken-ichi Suzuki, Tadao Nakamura
Graduate School of Information Sciences, Tohoku University,
6-6-01, Aramaki Aza Aoba, Aoba-ku, Sendai, 981-8579, Japan
{yukinori,suzuki,nakamura}@archi.is.tohoku.ac.jp

Abstract

The clustered architecture, where the conventional monolithic register file is partitioned into several smaller register files, is one of the candidates for the future high performance processor architectures. The aggressive partitioning can reduce the access time of the register file. On the other hand, the partitioning makes losses of instructions per clock cycle due to communication among register files. Not only the degree of partitioning, but the organization of partitioned register files also affects the access time of register files and the amount of inter-PE communication. In this paper, we investigate appropriate degrees of partitioning and organizations of partitioned register files in various 8-way issue clustered architectures. The results show that the eight single-issue PEs or the four double-issue PEs with the non-consistent register file organization can achieve the highest instructions per second and the speedup compared with the non-partitioned organization is 1.59 in the MediaBench suite and 1.69 in the SPEC2000int suite.

Keywords : clustered architecture, instruction-level parallelism, degree of partitioing, partitioned register file organization

1 Introduction

As the register file (RegFile) in modern high-performance processors is enlarged and multiported to support their wide issue policy, the access time for the RegFile becomes the critical path for clock cycle time and the energy of the RegFile accounts for a substantial portion of energy dissipation in processors [1, 2]. The enlarged and multiported RegFile makes its access time slower and the slow access time causes performance degradation. Moreover, the huge power consumption of an enlarged and multiported RegFile will become the biggest restriction of realizing future high-performance microprocessors[3]. The access time and energy of a RegFile is dependent on the number of registers and the number of ports. Reducing the number of registers and ports by means of partitioning a RegFile is one of the approaches to realize a reasonable RegFile.

Dynamically-scheduled clustered architectures, where the global structures are partitioned into simple smaller structures and each of them is arranged in a PE (processing element), will be able to realize a reasonable RegFile[4, 5, 6].

Here, a PE is also called a cluster in some papers. In clustered architectures, a conventional single monolithic RegFile is partitioned into several smaller RegFiles. The partitioning makes the access to the RegFiles faster because the number of entries and ports of the partitioned structures can be reduced. On the other hand, it increases the amount of inter-PE communication among PEs.

It is widely known that performance of clustered architectures depends on the amount of inter-PE communication to synchronize dependent instructions and the amount of workload imbalance that hinders parallel execution of instructions [5, 6]. The number of PEs and the issue width of each PE are important architectural parameters that balance the workload and communication among PEs. These architectural parameters also affect the clock cycle time of a processor and the energy dissipation of a RegFile because they determine the number of registers and ports.

In clustered architectures, the number of PEs is decided by degree of partitioning. The aggressive partitioning, where a single monolithic RegFile is partitioned into much more PEs, makes the RegFiles smaller and less ported. However, the aggressively larger number of PEs requires more communication among PEs. The issue width of each PE decides the number of ports of partitioned RegFiles. The wider issue width of each PE can process much more data without inter-PE communication in return for the slow multiported RegFiles. A smaller issue width of each PE can realize smaller and less ported RegFiles. However, it suffers from restricted ability for parallel processing within a PE, which induces the workload imbalance or inter-PE communication.

The other factor that determines the effectiveness of the partitioning is the organization of partitioned RegFiles. There are two organizations of partitioned RegFiles: multiple coherent RegFiles[7] and non-consistent RegFiles[8]. In the multiple coherent RegFiles, any register instances are replicated in each PE, so, every PE can utilize any register instances locally. In the non-consistent RegFiles, any register instances are not replicated, so, every PE can utilize only register instances in the local RegFile. The non-consistent RegFile organization can be realized by a smaller number of registers. However, register instances must be distributed to PEs properly, in this organization.

In this paper, we compare performance of various 8-way issue configurations with various degree of partitioning. At the same time, we also evaluate the effect of organizations of partitioned RegFiles. Finally we present the most appropriate design direction in terms of the total performance measured by the number of instructions per second.

The rest of this paper is organized as follows. In section 2, we briefly show the overview of a baseline clustered architecture and instruction steering scheme. Then, we discuss the effect of organization of partitioned RegFiles. Section 3 describes the experimental framework, the evaluation methodology and the results. Section 4 shows some related work. Section 5 concludes this paper.

2 Clustered architecture

2.1 Baseline microarchitecture

The microarchitecture of a clustered architecture is based on that of the aggressive out-of-order issue superscalar processors. Fig. 1 shows the overview

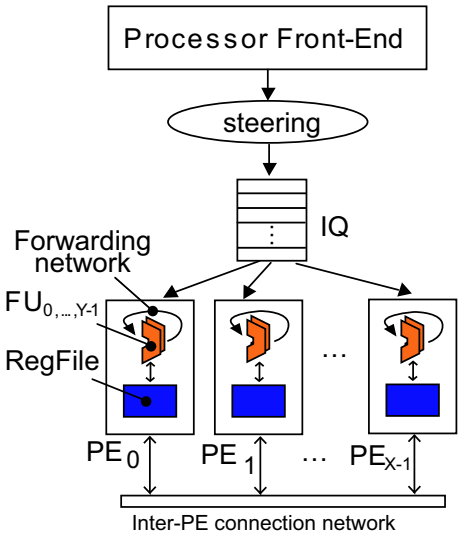

Figure 1: The overview of the baseline clustered architecture.

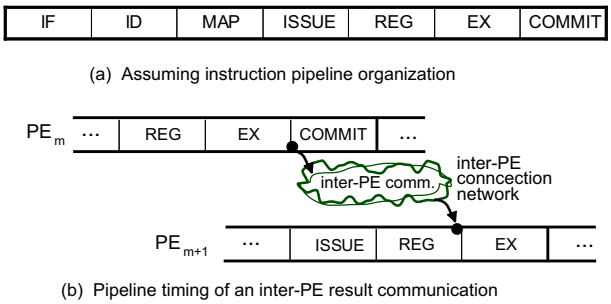

Figure 2: The timing of the pipeline.

of the baseline clustered architecture. The processor front-end fetches multiple instructions at once and decodes them. The decoded instructions are delivered to the steering logic. The steering logic chooses a PE for the execution of each instruction. Next, the steered instruction is dispatched to the IQ (issue queue) that observes whether the operand status of each instruction is ready or not. When required operands are ready, the instruction is waked up and the corresponding resources of the steered PE are checked. If the resources are available, the instruction is selected and issued to the PE and executed.

We represent a configuration of clustered architecture as X*Y, where X is the number of PEs and Y is the issue width in a PE. In this paper, we simulate the following four configurations as 8-way issue processors: 1*8, 2*4, 4*2 and 8*1. In the 1*8 configuration, any structures are not partitioned, so it corresponds to a conventional superscalar design. If the number of PEs is increased, faster data paths can be realized, however, data processing with the large number of PEs induces much more communication among PEs.

Fig. 2 (a) shows the pipeline stages assumed in this paper, which are based on those of Alpha21264 [7]. In MAP stage, instructions are dynamically steered

Table 1: The status of operands and its steered PE.

!ready scheme

in1 \ in2	null	!ready	ready
null	min_waiting	in2	min_waiting
!ready	in1	in1	in1
ready	min_waiting	in2	min_waiting

The min_waiting indicates that the instruction is steered to the PE with minimum waiting instrucitons.
The in1 and in2 indicate that the instruction is steered to the producer PE of source operand in1 and in2, respectively.

to appropriate PEs based on an instruction steering scheme and architectural registers specified by instructions are renamed to physical registers. The register renaming mechanism we adopted is also based on that of Alpha21264.

In ISSUE stage, instructions in the IQ are checked whether their operands are ready and their corresponding functional units are available. If the operands and the functional unit of an instruction is available, the instruction is issued and the unit is reserved for execution. After the operands are read in REG stage, the instruction is executed in EX stage in its given latency.

In the case where the instruction uses at least one unready operand, the instruction must wait until the results of the preceding instructions are provided. When the preceding dependent instruction is executed in the same PE as the waiting instruction, the waiting instruction is executed at the next cycle of the execution of the preceding instruction using a forwarding network. On the other hand, when the waiting instruction is allocated in a different PE from the preceding dependent instruction, the result from different PE must be transfered to the PE where the waiting instruction is allocated. We assume that it takes 2 extra cycles for this inter-PE communication as shown in Fig. 2 (b).

2.2 Baseline steering scheme

We use the !ready (not ready) instruction steering scheme as a baseline. This scheme steers instructions based on the status of operands and can achieve higher IPC [10]. In order to prevent the undesirable inter-PE communications, this scheme steers instructions with at least one unready operand to the same PE as its dependent instruction. Instruction without any unready operand is steered to the minimum loaded PE to improve the load balance among PEs. The status of operands is always monitored to realize the out-of-order execution in conventional processors. The heuristics for workload balancing is the number of waiting instructions in each PE because the min_waiting heuristics can obtain better performance than the DCOUNT heuristics[5].

Table 1 shows the relationship between the status of source operands of an instruction and its steered PE on the !ready scheme. The first column and first row of each table denote the status of the two source operands of a consumer instruction, in1 and in2. The status of a source operand is classified as follows: operand is nothing (null), a dependent operand is unready (!ready), or is ready

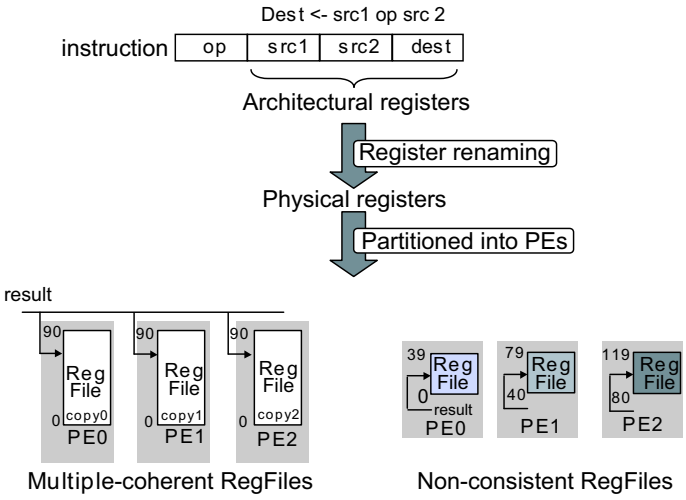

Figure 3: The register mapping process.

(ready). The rest of the table indicates which PE the instruction is steered to for each operand status. For example, if source operand in1 is ready and in2 is !ready, then the scheme steers the instruction to the same PE as the operand in2 is allocated.

In this paper, we assume all the PEs share a single large IQ as shown in Fig. 1 to isolate the problem of the lack of available IQ entries from our evaluation. In the case that we partition the IQ across the PEs, we have to take into account the utilization of each IQ. When the lack of available IQ entries occurs in a PE, an instruction cannot be steered into the PE, which might cause performance degradation. However, the workload balancing heuristics of the number of waiting instructions in each PE try to even up the utilization of each IQ, so the impact to IPC will be small. The effect of the partitioned IQ will be evaluated in our future work including considerations for the effect of wakeup delay on circuit level[4].

2.3 The organization of partitioned RegFiles

Most of current instruction set architectures are based on a single set of registers. However, the clustered architectures provide the physically partitioned set of registers in each PE. Therefore, a register mapping mechanism needs to map the architectural registers into the partitioned physical registers in an effective manner. Fig. 3 shows the register mapping processes in clustered architectures. Originally, source and destination operands of an instruction are specified by architectural registers. To resolve WAR and WAW dependencies, the architectural registers are renamed to the physical registers. There are two partitioned RegFile organizations to decide the mapping of architectural registers: multiple coherent register files and non-consistent register files. The biggest difference between them is whether an architectural register is mapped to multiple physical registers or one physical register.

In the multiple coherent RegFiles, an architectural register is mapped to all

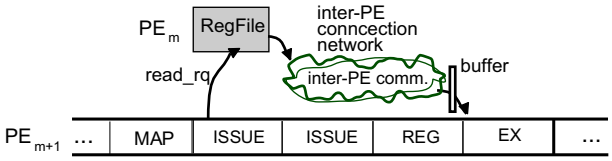

Figure 4: The pipeline timing of inter-PE register read in the non-consistent RegFile organizations.

of the partitioned RegFiles, so a register in each partitioned RegFile has the same register instance. An example of this organization is found in Alpha21264 [7]. The numbers on the left of the RegFiles in Fig. 3 indicate the identifiable numbers of the registers for register renaming. In the multiple coherent RegFiles, each register instance stored in the same position is always replicated. Therefore, this organization consumes the large number of registers to hold a copy of the register instance in each partitioned RegFile. The replication is done by writing all the produced results to the corresponding register of all the RegFiles, and this requires additional dedicated write ports for each RegFile.

In the non-consistent RegFiles, a register in each RegFile has its own register instance since a result is written into only one register where the result is produced [8]. This organization can reduce the total number of registers throughout the PEs because a register instance is not replicated. Therefore, as illustrated in Fig. 3, the identifiable numbers of registers are not replicated. Moreover, the non-consistent RegFiles can be realized only with the ports for the intra-PE functional units, plus a few ports to handle inter-PE communication.

On the other hand, in the non-consistent RegFile organization, register instances in each RegFile is different from each other, so inter-PE communication using the interconnection network is required if a PE is found to use non-local registers . Fig. 4 shows the pipeline timing of an inter-PE register read. We assume that the remote register read requires two cycles for communication compared with a single cycle when the operand is stored in the same PE. This communication delay due to the non-consistent RegFile organization might decrease the performance.

Not only the extra delay of register read, but also lack of available free registers in particular PEs might degrade the performance in the non-consistent RegFile organizations. This is because the number of available free registers in a PE is different from the others since the number of in-flight instructions in a PE is different from the others. When the PE selected by the instruction steering scheme lacks available free registers, the instruction must be reallocated to one of the other PEs, and this reallocation induces extra communication delay compared with the original PE allocation by the steering scheme.

The other difference between the multiple coherent and non-consistent RegFiles is the management of committed registers. In the renaming process of Alpha21264, committed registers are always stored in dedicated registers whose identifiable numbers are \$0-\$31. In the multiple coherent RegFile organization, those dedicated registers are also replicated across the partitioned RegFiles. On the other hand, in the non-consistent RegFile organization, we assume that the committed registers are partitioned into PEs and each PE has the same number of committed registers to avoid converging register pressure on a particular

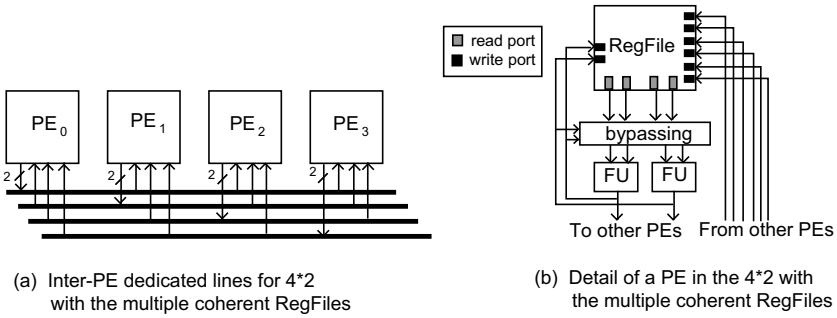

Figure 5: The multiple coherent RegFile organization.

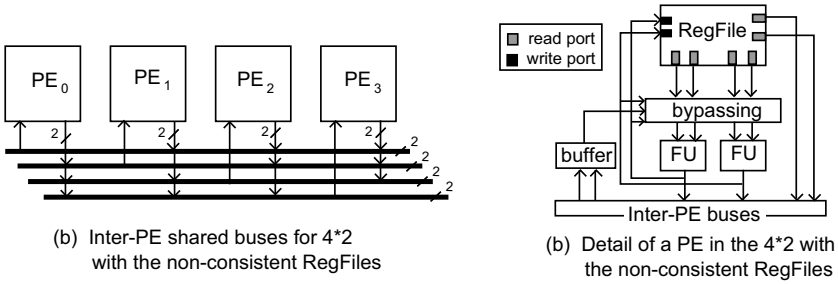

Figure 6: The non-consistent RegFile organization.

PE. For example, in an 8 PE configuration, 32 committed registers ($0-$31) are partitioned as follows: PE_0 has $0-$3, PE_1 has $4-$7, ... , PE_7 has $28-$31. In addition, in order to accommodate the renaming mechanism to the non-consistent organization, we prepare each PE its own free register list to handle the different numbers of available free registers among PEs.

2.4 The inter-PE network of the partitioned RegFiles

Partitioned RegFiles require inter-PE communication using an inter-PE communication network. The required inter-PE communication network is different depending on the organization of the partitioned RegFiles.

The multiple coherent RegFile organization requires a fully-connected network to deliver all the produced results for all RegFiles. To realize the network, dedicated signal lines and write ports for the result transfer are required. Fig. 5 (a) illustrates an overview of the inter-PE dedicated lines of the 4*2 with multiple coherent RegFile configuration. In this case, every PE can produce up to 2 results, so the required number of write ports for result coherence in a single PE is 6. Fig. 5 (b) illustrates details of a PE in the 4*2 configuration. In this case, total requirement for the ports in a PE is 4 reads and 8 writes.

The non-consistent RegFile organization provides shared buses for an inter-PE communication network as illustrated in Fig. 6 (a). Each shared bus corresponds to a write to one PE, and each PE is able to send data to any bus. There are various configurations for the shared buses unlike the dedicated lines of the multiple coherent RegFiles. We assume that the inter-PE communication network is made up of as many buses as the total issue width. Therefore, the

Table 2: The number of ports in each configuration

configuration	Multiple-coherent RegFiles	Non-consistent RegFiles
1*8	16R+8W	16R+8W
2*4	8R+8W	12R+4W
4*2	4R+8W	6R+2W
8*1	2R+8W	3R+1W

number of buses for a PE is the same number as the issue width in a PE. In the case of the 4*2 configuration, the total number of buses is eight and each PE has two buses for write to the PE. We also note that the maximum number of data transfered using shared buses per clock cycle in the non-consistent RegFile organization is the same as that of dedicated lines in the multiple coherent RegFile organization. This is because the maximum number of transfered data is set to the total number of the issue width for fair comparison.

Fig. 6 (b) depicts details of a PE in the 4*2 non-consistent RegFile configuration. We assume that the number of read ports in a RegFile for inter-PE network is the same as the issue width in a PE. In the case of the 4*2 configuration, up to 2 register instances from a single RegFile are allowed to read in a clock cycle for the input of the inter-PE network. We assume that when an operand from a remote PE is ready, the operand is send to the consumer PE as soon as possible. To support this communication, we assume a small buffer to save delivered operands from remote PEs and the operands remain in the buffer until the operand is consumed. We assume the size of the buffer will be small enough not to affect the operating frequency and the total amount of hardware cost[15]. Our model of the inter-PE communication network in the non-consistent RegFiles is similar to the model in [15].

This shared bus network for the non-consistent RegFiles can reduce the total number of ports in a partitioned RegFile. However, the shared bus network might cause resource conflicts of read ports or shared buses. In the case of a conflict, the communication is delayed and the waiting instruction is stalled until the data arrives.

The number of registers and ports are important parameters that decides the access speed and energy dissipation of RegFiles. In order to understand the relationship between the degree of partitioning and the organization of RegFiles, we summarize these parameters. For the number of registers, there are two metrics as follows: the number of net registers which excludes replicated registers, and the number of total physical registers which includes replicated registers. If we compare the two RegFile organizations with the same number of net registers, the multiple coherent RegFile organization requires the X times larger number of physical registers against the non-consistent RegFiles organization, where X represents the number of PEs.

The number of ports is also different depending on the partitioned RegFile organization. Table 2 shows the number of required ports for each organization. For the processing in a PE, we need 2Y read ports and Y write ports, where Y is the issue width in a PE. The number of ports for inter-PE communication network is (X-1)Y write ports for multiple coherent RegFiles and Y read ports for non-consistent RegFiles.

Table 3: Main architectural parameters.

Fetch and decode	8 instructions per cycle
Branch predictor	Tournament branch predictor
IQ, FQ, LQ, SQ size	64
ROB size	256
The number of total issues	8
Icache	128kB, 2way
Dcache	128kB, 2way

3 Experiments

3.1 Methodology

We developed a cycle-accurate execution-driven simulator to evaluate the various configurations of clustered architecture. Baseline simulator is sim-alpha [11], which is one of the extention versions of SimpleScalar tool set [12]. Sim-alpha models the detailed microarchitecture of Alpha21264, which is one of the clustered architectures composed of dual integer PEs (clusters) with multiple coherent RegFiles.

We modified sim-alpha to model the 8-way clustered architecture with all the architectural features including the degree of partitioning, the organizations of partitioned RegFiles and the number of registers per PE. The other architectural parameters are shown in Table 3. The rest of parameters such as latency of the caches and that of functional units are following that of Alpha21264.

We modeled the access time of partitioned RegFiles using CACTI-2.0 tool set [9] at 0.07 μm technology. Basically the model is intended to evaluate cache system, so we discarded the tag path of the model and set the width of a register to be 64 bits as depicted in [16].

We select a subset of 4 benchmarks (djpeg, cjpeg, rawdaudio, rawcaudio) from the MediaBench benchmark suite [13]. This benchmark suite captures the main features of commercial multimedia applications. Benchmarks which tend to achieve high instruction-level parallelism have been selected. We also select a subset of 7 benchmarks (gzip, vpr, gcc, mcf, perlbmk, bzip, twolf) from the SPEC2000CPU int benchmark suite [14]. The rest of SPEC2000 benchmarks could not be adapted to the simulation environment used. All the benchmarks were compiled for the Alpha binary using Compaq's C compiler v6.5 on Tru64 UNIX V5.1B with -O4 -fast -non_shared options. Each program of the MediaBench was executed until the completion and 100 million instructions of each program of the SPEC2000int were executed after forwarding 1 billion instructions.

3.2 Results

Fig. 7 shows instructions per clock cycle (IPC) for the MediaBench suite. The specifier following the colon in the legend of figure indicates the organization of the partitioned RegFiles. The 'MC' represents the organization with the multiple coherent RegFiles. The 'non-C' represents the organization with the

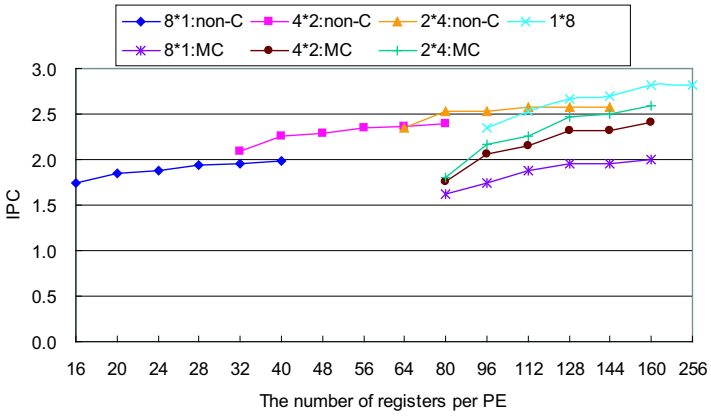

Figure 7: IPC of the MediaBench suite with various configurations.

non-consistent RegFiles. The x-axis represents the number of physical registers per PE.

The results show that IPC of 1*8 configuration is the highest of all. The 1*8 configuration does not partition any structures, so the 1*8 configuration can perform ideal IPC without any inter-PE communication losses. We can observe that the more RegFiles are partitioned, the lower the IPC becomes. This degradation is caused by inter-PE communication. It is also observed that IPC is increased when the number of registers per PE is increased in the same configuration. This is because the large number of registers can prevent IPC losses due to the lack of available free registers.

The net number of registers, which excludes replicated registers, is different depending on each partitioned RegFile organization. In the case that the number of net registers throughout PEs is equal, IPC of the MC organizations are higher than that of non-C organization. For example, comparing the 128 net registers case, IPC of the 8*1:MC 128 registers per PE configuration is higher than that of the 8*1:non-C 16 registers per PE configuration. The IPC losses of the non-C organization is caused by extra communication delay due to the inter-PE register read and the lack of available free registers in particular PEs.

It is hard for MC organizations to increase the net number of registers due to replicated registers. On the other hand, the non-C organization can increase the net number of registers at less cost. From the results, we can find out that when the number of register per PE is increased in the non-C organizations, they can achieve higher IPC than the corresponding MC 128 registers per PE organizations with the smaller number of total physical registers. Therefore, we can understand that the non-consistent RegFile organizations exploit the partitioned resources more effectively because it does not duplicate any register instances.

In return for the IPC losses due to the partitioning, the aggressive partitioning configuration can reduce the number of physical registers and the number of ports. Reducing the number of physical registers and ports enables faster access to RegFiles, which allows higher operating frequency of the processors. Fig. 8 shows the operating frequency of each configuration. The number of ports is

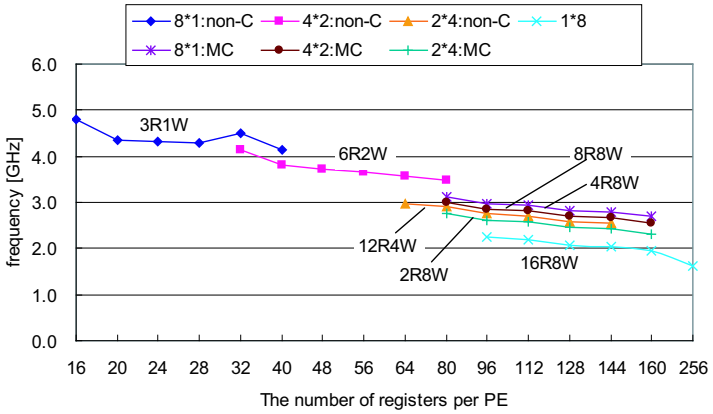

Figure 8: The operating frequency of the RegFiles.

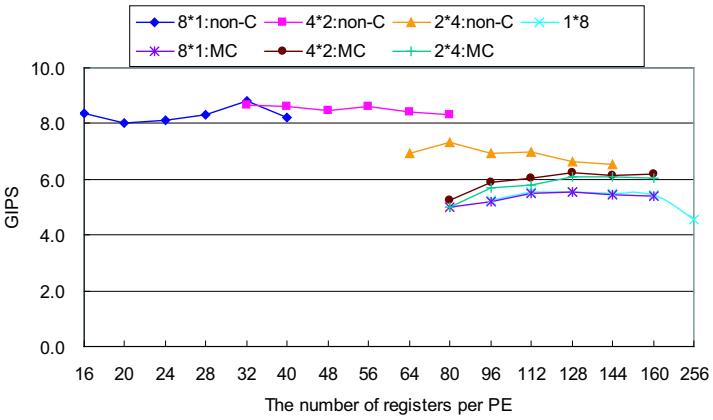

Figure 9: Giga Instructions Per Second of the MediaBench suite.

determined by the degree of partitioning and the organization of partitioned RegFiles, and the required number of ports is depicted in the figure.

We can understand that a RegFile is partitioned the more, the RegFile can achieve the higher frequency. This is because the number of ports is decreased by the partitioning and the smaller number of ports is effective in realizing the higher frequency. The smaller number of registers also makes RegFile access faster, however the benefit of the smaller number of registers is less than the smaller number of ports.

In terms of the organization of RegFiles, the operating frequency of the non-consistent RegFiles is superior than that of the multiple coherent RegFiles even in the same degree of partitioning. This is because the non-consistent RegFiles organization can reduce more ports. It is also observed that we do not partition any structure at all, the operating frequency becomes a half of the aggressive partitioning configuration with non-consistent RegFiles.

The increase of the number of PEs due to the aggressive partitioning requires

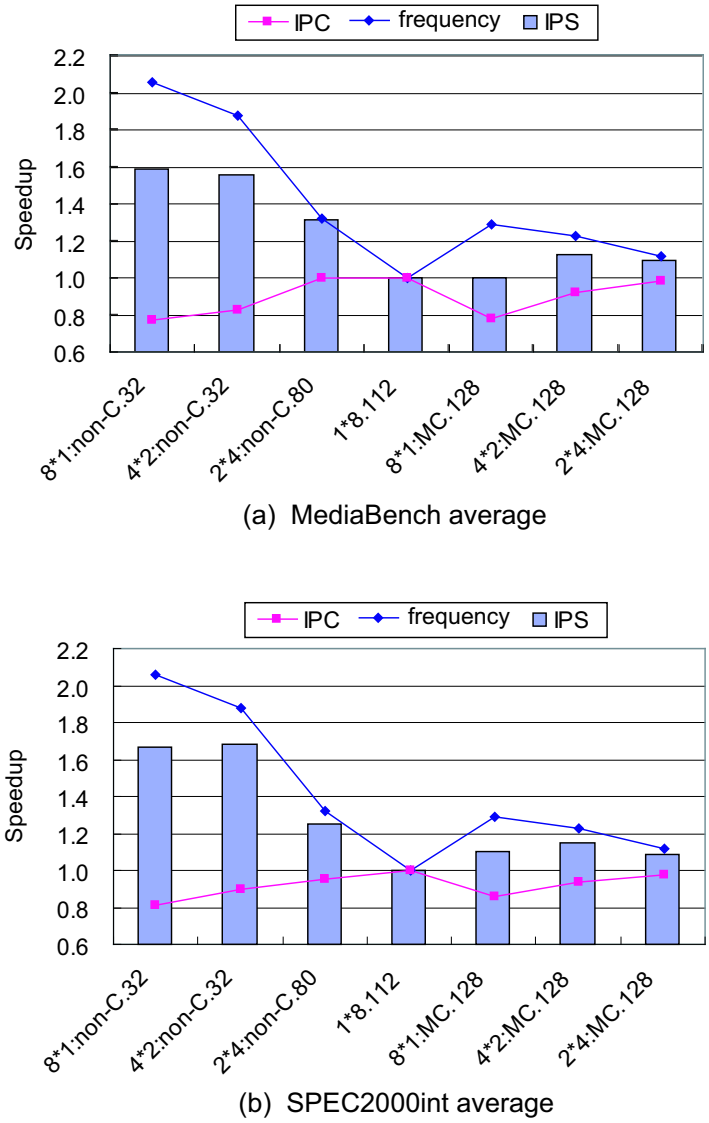

Figure 10: Speedup.

more communication among PEs, which causes the IPC degradation. On the other hand, the aggressive partitioning makes RegFile access faster and the faster RegFile allows higher operating frequency of the processor. Therefore, we introduce instructions per second metrics, which is calculated by multiplying IPC and operating frequency.

Fig. 9 shows GIPS (Giga Instructions Per Second) for the MediaBench suite. Here we assume that the operating frequency of a processor is equal to the operating frequency of the RegFile. From the results, we can understand that 8*1:non-C and 4*2:non-C configurations achieve higher IPS than any other configurations.

To illustrate the speedup due to the partitioning of RegFiles more clearly, we compare the peak IPS of each configuration. Fig. 10 shows the speedup ratio of IPC, operating frequency and IPS relative to the 1*8 configuration. We select the number of registers that can achieve the peak IPS for each configuration. The number following each configuration name in the figure represents the number of registers per PE.

The multiple coherent RegFile organizations cannot achieve high IPS due to the low operating frequency compared to the non-consistent organizations. The speedup of the configurations with the multiple coherent RegFile organization is around 10% in the 2*4:MC and 4*2:MC configurations and this is not so high compared with that of the non-consistent RegFile organizations. The IPC of 8*1:MC configuration is lower than the other multiple coherent RegFile configurations, so there is small speedup in 8*1:MC configuration compared with the baseline configuration.

On the other hand, it is remarkable that the 8*1:non-C configuration achieves 1.59 times higher IPS than the baseline 1*8 configuration in the MediaBench suite and the 4*2:non-C configuration achieves 1.69 times higher IPS in the SPEC2000int suite. The reason of these IPS gains is that the non-consistent RegFile organization overcome the disadvantage of lower IPC using the benefit of its higher frequency. Therefore, we can summarize that the aggressive partitioning configurations with the non-consistent RegFile organization is the most efficient design that balances both the inter-PE communication losses and the faster RegFile access.

4 Related Work

In order to obtain higher performance in dynamically-scheduled clustered architectures, there are many proposals for instruction steering schemes and their comparisons in literature [5, 6, 10]. However, degree of the partitioning and an organization of the partitioned RegFiles are also the other factors that determine performance of clustered architectures. The trade-off of degree of the partitioning between the lower IPC and the faster data paths has not been evaluated. Zyuban and Kogge evaluate the degree of the partitioning in terms of energy efficiency[15]. However, they do not discuss the trade-off between the lower IPC and the faster data paths and they only focused on the non-consistent RegFile organizations.

Many proposals of Dynamically-scheduled clustered architectures assume the multiple-coherent RegFile organization [7, 6]. Our work is the first work in terms of comparison of the multiple-coherent RegFile and non-consistent RegFile organization. Our model of the non-consistent RegFiles is similar to the model in [15]. The model in [15] assumes a remote access buffer (RAB) to feed data to remote PEs. In contrast, our model assumes a buffer to save data from remote PEs. Intrinsically, these two timing model are the same.

The partitioned RegFile model in [5] inserts copy instructions between dependent instructions dynamically. The required register instances are replicated across PEs, so this can be seen as partially non-consistent RegFiles. This timing model of processing dependent instructions is the same as that of our the non-consistent RegFiles.

The partially non-consistent RegFile organization in [5] does not require the

extra buffer for inter-PE communication network. However, this organization must add the extra write ports to receive data from remote PEs and the more registers to hold the copied instances. For the non-consistent RegFile organization, Zyuban et al. mentioned that 6 entries in the remote access buffer is sufficient to save data for inter-PE communication and this will be small enough not to degrade the operating frequency[15]. Considering these facts, we assume using the additional buffer. The comparison between the partially non-consistent and fully non-consistent RegFiles will be our future work.

Seznec and Rochecouste proposed register Write Specialization and register Read Specialization for clustered architecture [16]. Based on the multiple coherent RegFile organization, they force functional units in a PE to write and read the specific registers. The register write specialization enables the number of ports of registers to reduce. The register read specialization can reduce the number of replicated registers. This RegFile organization is referred as the multiple coherent write specialization RegFile organization.

Brown and Patt evaluated performance of multiple coherent write specialization RegFile organization and partially non-consistent RegFile organization[17]. They concluded that organization with the write specialization RegFiles can achieve about 10% higher IPC than that with the partially non-consistent RegFiles. However, they did not vary the number of registers in the partially non-consistent RegFiles. In this paper, we can find out that if we adjust the number of registers, the non-consistent RegFile organization can achieve higher IPC than multiple coherent RegFile organization.

5 Conclusions

In this paper, we have compared performance of various 8-way issue configurations to investigate appropriate degree of partitioning and organizations of partitioned register files. The partitioning of a single monolithic register file into several smaller ones makes the register file access time shorter because the number of entries and ports can be reduced. On the other hand, the partitioning increases the amount of inter-PE communication among PEs and this increase of communication causes IPC degradation. The organizations of partitioned register files also have an impact on performance because the amount of inter-PE communication, the number of registers and ports differs from each organization.

We have observed that IPC is decreased as the the number of PEs is increased. This is because the partitioning requires more communication among PEs. On the other hand, the more aggressive partitioning makes the higher operating frequency. We also have introduced instructions per second metrics to investigate the best design. The results show that the aggressive partitioning configurations with the non-consistent register file organization can achieve the highest instructions per second and the speedup compared with the non-partitioned configuration is 1.59 in the MediaBench suite and 1.69 in the SPEC2000int suite.

References

[1] Il Park, Michael D. Powell, and T. N. Vijaykumar. Reducing register ports for higher speed and lower energy. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, pp. 171–182, 2002.

[2] Rajeev Balasubramonian, Sandhya Dwarkadas, and David H. Albonesi. Reducing the complexity of the register file in dynamic superscalar processors. In *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture*, pp. 237–248. IEEE Computer Society, 2001.

[3] V. Zyuban and P. Kogge. The energy complexity of register files. In *Proceedings of the 1998 international symposium on Low power electronics and design*, pp. 305–310, 1998.

[4] Subbarao Palacharla, Norman P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *Proceedings of the 24th annual international symposium on Computer architecture*, pp. 206–218, 1997.

[5] Joan-Manuel Parcerisa and Antonio González. Reducing wire delay penalty through value prediction. In *Proceedings of the 33rd annual international symposium on Microarchitecture*, pp. 317–326, 2000.

[6] Aneesh Aggarwal and Manoj Franklin. An empirical study of the scalability aspects of instruction distribution algorithms for clustered processors. In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 172–179, 2001.

[7] R. E. Kessler. The alpha 21264 microprocessor. *IEEE Micro*, Vol. 19, No. 2, pp. 24–36, 1999.

[8] Josep Llosa, Mateo Valero, and Eduard Ayguade. Non-consistent dual register files to reduce register pressure. In *Proceedings of the 1st IEEE Symposium on High-Performance Computer Architecture*, pp. 22–31, 1995.

[9] Glen Reinman and Norman P. Jouppi. CACTI 2.0: An integrated cache timing and power model. Technical report, WRL Research Report 2000/7, 2000.

[10] Yukinori Sato, Kenichi Suzuki, and Tadao Nakamura. An operand status based instruction steering scheme for clustered architectures. In *Proceedings of the 2005 International Conference on Computer Design (CDES'05)*, pp. 168–174, 2005.

[11] Rajagopalan Desikan, Doug Burger, and Stephen W. Keckler. Measuring experimental error in microprocessor simulation. In *Proceedings of the 28th annual international symposium on Computer architecture*, pp. 266–277, 2001.

[12] Doug Burger and Todd M. Austin. The simplescalar tool set, version 2.0. *Compututer Architecture News*, Vol. 25, No. 3, pp. 13–25, 1997.

[13] Chunho Lee, Miodrag Potkonjak, and William H. Mangione-Smith. MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, pp. 330–335, 1997.

[14] John L. Henning. SPEC CPU2000: Measuring CPU Performance in the new millennium. *IEEE Computer*, Vol. 33, No. 7, pp. 28–35, 2000.

[15] Victor V. Zyuban and Peter M. Kogge. Inherently lower-power high-performance superscalar architectures. *IEEE Transactions on Computers*, Vol. 50, No. 3, pp. 268–285, 2001.

[16] Andre Seznec, Eric Toullec, and Olivier Rochecouste. Register write specialization register read specialization: a path to complexity-effective wide-issue superscalar processors. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, pp. 383–394, 2002.

[17] Mary D. Brown and Yale N. Patt. Demand-only broadcast: Reducing register file and bypass power in clustered execution cores. In *Proceedings of the First Watson Conference on Interaction between Architecture, Circuits, and Compilers*, 2004.