EXPLORING THE MEMORY HIERARCHY FOR PACKET PROCESSING APPLICATIONS

Bahareh Safaei K.[1], Ensiyeh S. F. Moghaddam[2], Mona Raeesi[2] , and Mehdi Alipour[2]

[1] Dept of Electrical, computer and IT engineering, Qazvin Islamic Azad University,Qazvin 34185-1416 Iran.
[2] Allameh Rafiei Highr Education Institute of Qazin, Iran
Mehdi.alipour@qiau.ac.ir, mehdi_10f@yahoo.com

ABSTRACT

In the future, network processors must process more computation-intensive network applications and internet traffic and packet-processing tasks become heavier and sophisticated. Since the processor performance is severely related to the average memory access delay and also the number of processor registers affects the performance, cache and registerfile are two major parts in designing embedded processor architecture. Although increasing cache and registerfile size leads to performance improvement in packet processing tasks in high traffic networks with too much packets, the increased area, power consumption and memory hierarchy delay are the overheads of thesetechniques. Therefore, implementing these components in the optimum size is of significant interest in the design of embedded processors. This paper explores the effect of cache and registerfile size on the processor performance to calculate the optimum size of these components for network applications. Experiment a results show that although having bigger cache and registerfile is one of the performance improvement approaches in network processors, however, by increasing the size of these parameters over a threshold level, performance improvement is saturated and then, decreased.

KEYWORDS

Network processor, design space exploration, cache, optimum size of registerfile, cache access delay.

1. INTRODUCTION

In recent years network application and internet traffic become heavier and sophisticated so, future network processors will be encountered by more computation-intensive network applications, in this way, designing high performance processors is recommended. By scaling down the technology and presentation of chip multiprocessors (CMP) that are usually multi-thread processors, somehow the user's performance necessity have guaranteed. Inseparable parts in designing these processors are cache and registerfile because the performance of a processor is severely related to cache access and also having enough registers.

Recently in numerous research, multi-thread processors are used to design a fast processor especially in network processors [4], [9], [11], [23], [25] ,[26]. In [3] a Markov model based on fine grain multithreading is implemented. Analytical Markov model is faster than simulation and has dispensable inaccuracy. In this chain, stalled threads defined as states and transitions are based on cache contention between threads.

Cache memories are usually used to improve the performance and power consumption by bridging the gap between the speed and power consumption of the main memory and CPU, therefore, the system performance and power consumption is severely related to the average

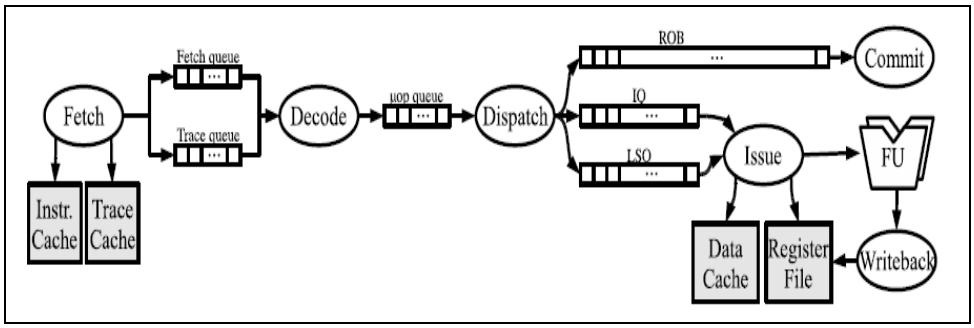

Fig1. Processor pipeline of Multi2sim simulator[19].

memory access time and power consumption which makes cache as a major part in designing embedded processor architectures. In [4] cache misses are introduced as a factor for reducing memory level parallelism between threads. In [5] thread criticality prediction has been used and for better performance, resources are given to the threads that have higher L2 cache misses which are called the most critical threads. To improve packet-processing in network processors, [6]-[8] have applied direct cache access (DCA). In [9] processor architecture is based on the simultaneous multithreading (SMT) and cache miss rate is used to show the performance improvement.To find out the effect of cache access delay on performance, a comparison between multi-core and multi-thread processors has been performed in [10]. Likewise, victim cache is an approach to improve the performance of a multi-thread processor [11].

All recent researches are based on the comparison results with single-core single-thread processors. In the other word multi-thread processors are the heir of the single thread processors [23] ,[25] ,[26]. Hence, evaluating the effective parameters like cache and registerfile sizes are required for designing a multithread processor. The basic purpose of this paper is to study the effect of the cache size on the performance because network processors process computation and data intensive applications and larger cache sizes will present better performance.

Generally, one of the easiest way to improve the performance of embedded and network processors is increasing the cache size [2], [12] ,[13], [14], [22]-[26] but this improvement, severely increase the occupied area and power consumption of the processor. So, it is necessary to find a cache size that creates tradeoffs between performance and power-area of the processor.

From other point of view, because of the performance per area parameter, higher performance in a specified area budget is one of the most important needs of a high performance embedded processor. A negative point of the recent researches is that they don't have any constraints on the cache size. Because of the limited area budget in embedded processors, in this paper we have found the optimum size of L1 and L2 cache and also, because of the longer latency of bigger caches, best size of memory hierarchy in relation to this parameter has calculated.

As mentioned above, another inseparable part in designing embedded processors is registerfile. Like the cache, size of this parameter has the fundamental effect on the better processor performance. To improve the performance of a network processor, a large registerfile must be implemented. However, larger registerfiles occupy more area and make a worse critical path [18]. Therefore, exploring the optimum size of the registerfile is the second purpose of this paper.

The high importance of this issue is based on the fact that some parameters encourage designer to have a large registerfile.

Generally network processors, are implemented in multi-issue architecture and out of order (OOO) instruction execution that has renaming logic [16]-[18], [23], [25], [26]. On the other hand, because registerfiles are shared in multi-thread processors, observing the fact: increment the common parts in design, force the designer to have a larger registerfile [1]. These parameters also make higher importance for registerfile size.

In [15] effects of registerfile size in SMT processors have been studied. However, high budget for the number of registers has used. In recent researches effect of registerfile and cache size in the same time is not studied, so in this paper this issue will be studied too.

2. SIMULATION ENVIRONMENT

For simulation, we have used Multi2sim version 2.3.1 [19], a superscalar multi-thread multi-core simulation platform which has 5 stages of pipeline named *fetch, decode, dispatch, issue, writeback, and commit*. and executes x86 instruction set architecture. Fig.1 shows a block diagram of the processor pipeline modeled in Multi2Sim.In the fetch stage, instructions are read from the instruction or the trace cache. Depending on their origin, they are placed either in the fetch queue or the trace queue. The former contains raw macroinstruction bytes, while the latter stores predecoded microinstructions (uops). In the decode stage, instructions are read from these queues, and decoded if necessary. Then, uops are placed in program order into the uop queue. The fetch and decode stages form the front-end of the pipeline [19].

The dispatch stage takes uops from the uop queue, renames their source and destination registers, and places them into the reorder buffer (ROB) and the instruction queue (IQ) or load-store queue (LSQ). The issue stage is in charge of searching both the IQ and LSQ for instructions with ready source operands, which are schedule to the corresponding functional unit or data cache. When and uop completes, the writeback stage stores its destination operand back into the register file. Finally, completed uops at the head of the ROB are taken by the commit stage and their changes are confirmed. So the commit stage is where we can log and count the number of committed instructions for performance comparison. Detail of this simulation will be described in part IV, simulation method and results.

Multi2sim can run programs in multi issue platform, but to evaluate the requirements of each thread we have used the single issue model for comparison. We changed and compiled the source code of simulator on a 2.4GHz, dual core processor with 4GB of ram and 6MB of cache that run fedora 10 as an operating system. Base on this configuration the average time of each simulation is about 20 minutes.

3. BENCHMARK

We used packetbench [20] for extracting our results. This benchmark is a good platform to evaluate the workload characteristics of network processors. It reads and write packets from and to real packet traces, and manage packet memory, and implement a simple application programming interface API. This involves reading and writing trace files and placing packets into the memory data structures used internally by PacketBench. On a network processor, many of these functions are implemented by specialized hardware components and therefore should not be considered part of the application. Programs in this tool are categorized in 3 parts : *1- IP forwarding* which is corresponding to current internet standards. *2- packet classification* which is commonly used in firewalls and monitoring systems. *3-encryption,* which is a function that actually modifies the entire payload

TABLE 1. number of instructions, macro-instructions and cycles of main function of each application.

	Ipv4_lctrie	Ipsec	Flow_class
# of ins.	203.42	94815.41	105.47
# of macro_ins.	327.54	162696.31	225.07
# of cycles	1096.32	166992.01	1075.02

of the packet. Specific application that respectively we used from each category are IPv4-Lctrie, Flow-Classification and IPSec.

Pv4-trie performs RFC1812-based packet forwarding. This implementation is derived from an implementation for the Intel IXP1200 network processor. This application uses a multibit trie data structure to store the routing table, which is more efficient in terms of storage space and lookup complexity [20].

Flow classification is a common part of various applications such as firewalling, NAT,and network monitoring. The packets passing through the network processor are classified into flows which are defined by a 5-tuple consisting of the IP source and destination addresses, source and destination port numbers, and transport protocol identifier. The 5-tuple is used to compute a hash index into a hash data structure that uses link lists to resolve collisions[20].

IPSec is an implementation of the IP Security Protocol[20] , where the packet payload is encrypted using the Rijndael Advanced Encryption Standard (AES) algorithm [9]. This is the only application where the packet payload is read and modified.

4. SIMULATION METHOD AND RESULTS

Purpose of this paper is to evaluate optimum size of cache and registerfile. At first, we describe the methodology to extract proper size of cache. For this purpose, it is necessary to configure the simulator in the way that just the size of cache be the parameter that has affects on the performance. So, for each application the execution number of the main function is calculated in different sizes of L1 and L2 caches. For this purpose we made changes in some parts of simulator source code to calculate the cycles of sending a packet (the cycles that are used to execute the main function of each application).

To calculate the beginning address and end address of the main function we disassemble the executable code of each benchmarks and extract these addresses and then this parameters back annotated to commit.c and processor.h file of Multi2sim simulator where a thread is executed.

TABLE 2
THE MOST IMPORTANT PARAMETERS USED IN CACTI

	L1 cache	L2 cache
Cache size	Variable	Variable
Cache line size	Variable	Variable
Associatively	Variable	Variable
Number of banks	1	1
Technology node (nm)	90nm	90nm
Read/write ports	1	1
Exclusive read ports	0	0
Exclusive write ports	0	0
Change tag	No	No
Type of cache	Fast	normal/serial
Temperature (K)	300-400	300-400
RAM cell/transistor type in data array	ITRS-HP	Global
RAM cell/transistor type in tag array	ITRS-HP	Global

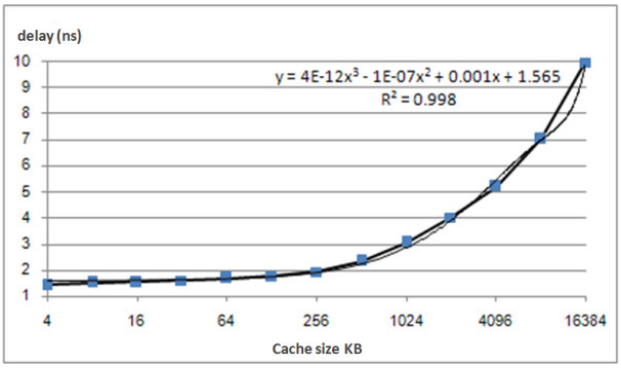

Fig.2 effect on cache size on cache access delay.

By these changes we can calculate the number of x86 instructions and macroinstructions and the cycles for specific function. We gathered the simulation results of processing 10000 packets and the average is illustrated in table 1. Second step is to run the simulator with different cache sizes. But the worthwhile point is that although based on the recent researches that recommend doubling the cache size for improving the performance of a processor, however during doubling the cache size, important parameters like area power and cache access delay must be considered. For this purpose we have used CACTI 5.0 [21], a tool from HP that is a platform to extract parameters relevant to cache size considering fabrication technology. Most important parameters that we used in this research are in table 2.

To compare the performance based on the cache size, extracted results from cacti (L1 and L2 cache access delay) are back annotated to Multi2sim. This work has been done by calculating the simulator cycle time and comparing it to the results of cache access time from CACTI. In this

way when the cache size is changed, actual cache access delays are considered. As can be seen in Fig.2, increasing the cache size, leads to more cache access delays.

For exploring the cache size, the other simulator parameters are set to the default value, because the purpose is to find the best cache size for a single-thread single-core processor for network applications. i.e. width of the pipeline stages must be one (issue-width =1).

5. ANALYSIS OF THE SIMULATION RESULTS

Fig.3 shows the extracted result of our simulations. In this figure each axis has labled and the vertical axis (per-pen) shows the performance penalty of related cache size configuration. Based on these results, by increasing the cache size we can achieve more hit rates, however, because of the longer cache access time of larger caches, from a specific point (best cache size) performance improvement is saturated and then even decreased. In another word, doubling the cache size always cannot improve the performance.

From another point of view, area budget is limited and always we can't have a large cache, so, considering the sizes smaller and near the best cache size, performance degradations are negligible (3% in average). To calculate the optimum size of registerfile, we have applied the parameters used for calculating best cache size, however, to find out just the effect of registerfile size on the performance, we used the best cache size (L1 and L2) concluded in the previous section for the cache size and run the simulator accordingly. Fig.4 shows the results of this part. In this picture vertical column shows the performance effect(performance penalty) of registerfile size. Numbers in this column are relative to the best size of registerfile. It shows that although for all applications in average the best size of registerfile is 68 and above but in sizes near the half of this size performance penalty is lower that 5%. Also this figure shows that reducing the registerfile size always decrease the performance but sometimes, by doubling the

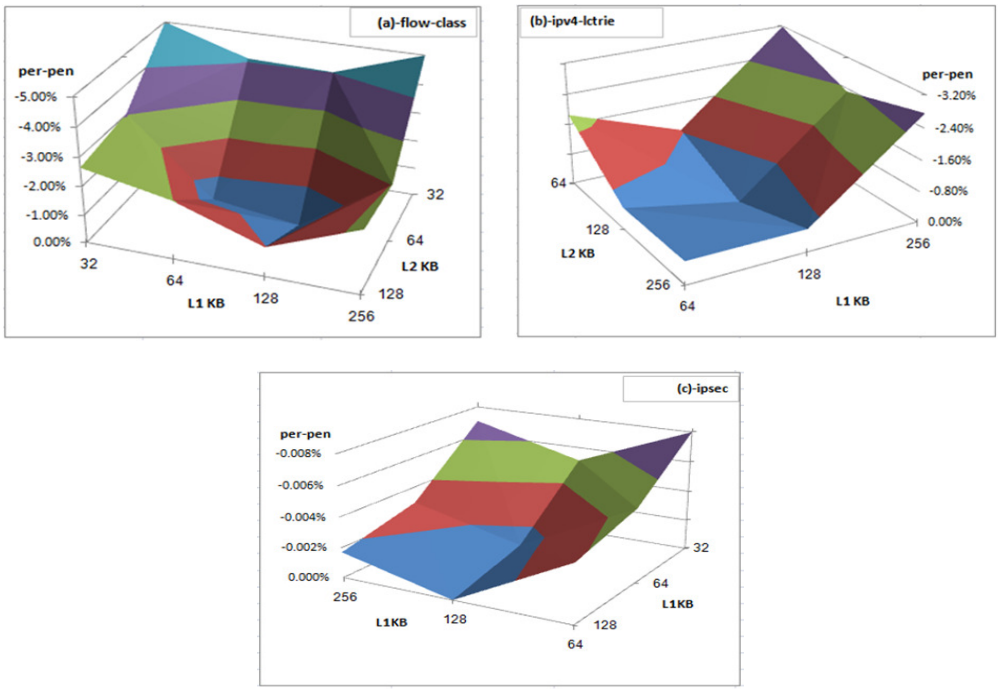

Fig. 3 effect of cache size on the performance (a):Flow_class, (b): IPV4lctrie, (c):IPsec

registerfile size we don't have noticeable performance improvement. So the first point that the highest performance is met, is introduced as the best size for registerfile.

It is worthwhile to say that in Fig.4 the concurrent effect of cache size and registerfile size can be seen, something that isn't mentioned in the recent researches. In another side of view based on recent researches [23,25,26] to have a multi-thread architecture we need more area budget, and to run this architecture in the best performance that can be met, multi issue architecture with renaming logic , ROB, LSQ, IQ and other OOO components which occupy large area budget are needful.Base on these simulations, we calculated 2 point for cache and registerfile sizes : *1- best size* that has no performance penalty and occupy bigger area budget and *2- optimum size* that has about 3% performance penalty and occupy smaller area budget so, we can deduce that in the optimum size of cache and registerfile we have saved the area budget of the processors. andqualification to run multi-threads in the higher isuue-widhts is obtained. In another word in lower area more performance is achieved and cause to growth the most important parameters for embedded applications that is performance per area. As mentioned before the multi-thread processors are the heir of single thread processors. So, extracting the best size for important parameters like cache size and registerfile size is necessary.

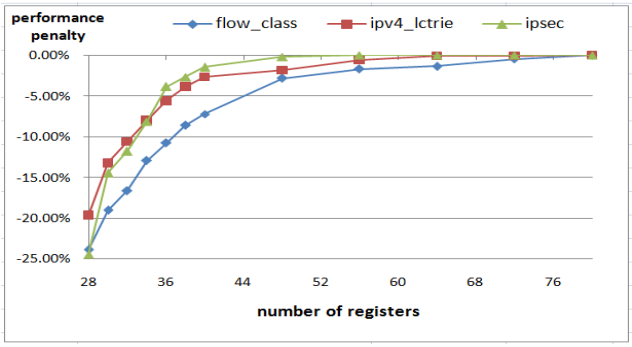

Fig.4 effect of registerfile size on performance

6. CONCLUSION

In this paper we studied the effect of cache and registerfile size on the performance, and extracted the best size of these two parameters for network applications. Simulation results show that for selected benchmarks best size of L1 and L2 caches are 64KB and 128KB respectively and the best size of registerfile is 80. Experiments show that although by increasing the cache size performance will improve, but in a specific point the performance improvement is saturated and then decreased. Also increasing registerfile size always cannot improve the performance and in a specific size the performance improvement will be saturated. From the area point of view, based on the results of this research, when we select half of the best size of the cache and registerfile, performance penalty is about 3% in average. In another word in sizes lower than best size the acceptable performance can be met. It means in the in lower area we can reach performance needs and have a beter performance/area parameter.

REFERENCES

[1] David A. Patterson, John L. Hennessy.: Computer *organization and design*: the hardware/software interface, Morgan Kaufman 2007.

[2] Davanam, N., Byeong Kil Lee., "Towards Smaller-sized Cache for Mobile Processors Using Shared Set-Associatively," international conference on information technology. pp. 1-6. 2010.

[3] Chen, X.E., Aamodt, T.M., "A First-Order Fine-Grained Multithreaded Throughput Model," International Symposium on High Performance Computer Architecture(HPCA), pp.329-340.2009.

[4] S hailender, Chaudhry. Robert, Cypher. Magnus, Ekman. Martin, Karlsson. Anders, Landin. Sherman, Yip. Haakan, Zeffer. Marc,Tremblay.," Simultaneous speculative threading: a novel pipeline architecture implemented in sun's rock processor,"internationalsymposium on computer architecture(ISCA 2009). pp.484-495. 2009.

[5] Abhishek, Bhattacharjee. Margaret, Martonosi. "Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors,"International Symposium on Computer Architecture (ISCA). pp.168-176. 2009.

[6] Huggahalli, R. Iyer, R. Tetrick, S.,"Direct cache access for high bandwidth network I/O," International Symposium on computer Architecture (ISCA). pp.50-59.2005.

[7] Kumar, A. Huggahalli, R.,"Impact of Cache Coherence Protocolson the Processing of Network traffic,"International symposium on Microarchitecture. (MICRO),pp.161-171. 2007

[8] Kumar, A. Huggahalli, R. Makineni, S.,"Characterization of Direct Cache Access on Multi-core Systems and 10GbE," International Symposium on High Performance Computer Architecture(HPCA). pp.341-352. 2009.

[9] Kyueun, Yi. Gaudiot, J.-L.,"Network Applications on Simultaneous Multithreading Processors," Journal of IEEE Transaction on Computer (TCOMPUTER). pp.1200-1209. 2009

[10] Guz, Z. Bolotin, E. Keidar, I. Kolodny, A. Mendelson, A. Weiser, U.C.," Many- Core vs. Many-Thread Machines: Stay Away From the Valley," Journal Computer Architecture Letters (L-CA),pp.25-28. 2009.

[11] Colohan, C.B. Ailamaki, A.C. Steffan, J.G. Mowry,T.C.,"CMP Support for Large and Dependent Speculative Threads," Journal IEEE Transaction on Parallel and Distributed systems (TPDS) , pp.1041-1054, 2007.

[12] Bienia, C. Kumar, S. Kai Li.," PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip- Multiprocessors," International Symposium on WorkloadCharacterization (IISWC) , pp.47-56. 2008.

[13] Guanjun, Jiang. Du, Chen. Binbin, Wu. Yi, Zhao. Tianzhou, Chen. Jingwei, Liu.," CMP Thread Assignment Based on Group sharing L2 Cache," International Conference on Embedded Computing .pp. 298-303. 2009.

[14] McNairy, C. Bhatia, R.," Montecito: a dual core dual thread Itanium processor," IEEE Journal MICRO , pp.10- 20. 2005.

[15] Alastruey, J. Monreal, T. Cazorla, F. Vinals, V. Valero, M.,"Selection of the Register File Size and the Resource Allocation Policy on SMT Processors Policy on SMT Processors," International Symposium on Computer Architecture and High Performance Computing (SBAC- PAD), pp.63-70. 2008.

[16] A, Yamamoto. Y, Tanaka. H, Ando. T, Shimada.," Data prefetching and address pre-calculation through instruction pre-execution with two-step physical register deallocation," in MEDEA-8. pp.41–48. 2007.

[17] Yamamoto. Y, Tanaka. H, Ando. T, Shimada.,"Two-step physical register deallocation for data prefetching and address precalculation," IPSJ Trans. on Advanced Computing Systems. vol. 1. no. 2. pp. 34–46. 2008.

[18] Tanaka, Y. Ando, H.,"Reducing register file size through instruction pre execution enhanced by value prediction," IEEE International Conference on Computer Design .pp. 238 – 245. 2009.

[19] R, Ubal. J, Sahuquillo. S, Petit. P, L'opez.," Multi2Sim: A SimulationFramework to Evaluate Multicore-Multithreaded Processors," journal Proc. of the 19th Int'l Symposium on Computer Architecture and High Performance Computing. Oct 2007.

[20] Ramaswamy, Ramaswamy. Tilman, Wolf.," PacketBench: A tool for workload characterization of network processing," in Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6), Austin, TX. pp. 42- 50. Oct. 2003.

[21] Shyamkumar Thoziyoor, Naveen Muralimanohar, and Norman P. Jouppi "CACTI 5.0 technical report", form Advanced Architecture Laboratory, HP Laboratories HPL- 2007. Online].Available:www.hpl.hp.com/research/cacti/

[22] Hyunjin, Lee. Sangyeun, Cho. Childers, B.R.," StimulusCache: Boosting Performance of Chip Multiprocessors with Excess Cache," IEEE 16th International Symposium on High Performance Computer Architecture (HPCA). pp. 1 – 12. 2010.

[23] Chung, E.S. Hoe, J.C.," High-Level Design and Validation of the BlueSPARC Multithreaded Processor," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD). Vol.29. Issue: 10. pp. 1459 – 1470. 2010.

[24] Davanam, N. Byeong, Kil. Lee.," Towards Smaller-sized Cache for Mobile Processors Using Shared Set- Associativity," international conference on information technology, pp. 1-6, 2010.

[25] Kyueun Yi, and Gaudiot J. L, "Network Applications on Simultaneous Multithreading Processors ,"*IEEE Transaction on Computer(TCOMPUTER).*VOL.59,NO. 9, pp.1200- 1209, SEPTEMBER 2010.

[26] Wang, H. Koren, I. Krishna, C.," Utilization-Based Resource Partitioning for Power-Performance Efficiency in SMT Processors," IEEE Transactions on Parallel and Distributed Systems,(TPDS) Vol. 22, no. 99, pp. 191-216, 2010.