行政院國家科學委員會專題研究計畫　期中進度報告

嵌入系統中低功率快取記憶體結構之設計(1/2)

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-002-064-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立臺灣大學資訊工程學系暨研究所

計畫主持人：楊佳玲

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 92 年 5 月 30 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※

※

※　　嵌入系統中低功率快取記憶體結構之設計

※　　　　　　　　　　　　　　　　　　　　※

※

※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：x 個別型計畫　　□整合型計畫

計畫編號：NSC　92-2213-E-002 –014-

執行期間：　　91 年　8 月　1　日至　92 年　7　月 31 日

計畫主持人：楊佳玲

共同主持人：

計畫參與人員：曾宏偉 楊善詠 辛逸軒 李建豪

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：台灣大學資訊工程學系

中　華　民　國　92　年 5　月 30 日

行政院國家科學委員會專題研究計畫成果報告

嵌入系統中低功率怪快取記憶體結構之設計

Designs of Energy-Efficient Cache for Embedded Systems

計畫編號：NSC 92-2213-E-002 –014-

執行期限：91 年 8 月 1 日至 92 年 7 月 31 日

主持人：楊佳玲　　台灣大學資訊工程系

e-mail: yangc@csie.ntu.edu.tw

http://www.csie.ntu.edu.tw/~yangc

一、中文摘要

由於可攜式裝置（如：手持式電腦與個人通訊設備）的普遍性日益增加，電源消耗成為一個必要的設計考量。快取記憶體在晶片的整體電源消耗上佔有不可忽視的成分。在這個計劃裡，我們提出一個新的快取記憶體結構，它能利用應用程式中資料存取的特色來達到電源延遲最佳化。一個常被使用來節省快取記憶體耗電的技術，是盡量減小快取記憶體被存取的範圍。然而，此種減少電源消耗的技術通常也會犧牲部分的效能，因為我們無法正確地預測所需要的資料存在於快取記憶體中的何處。我們在此計劃中提出的快取記憶體結構，利用了軟硬體互相搭配的機制，來分配程式中不同型態的資料到快取記憶體中不同的部分。藉由控制快取記憶體資源之分配，我? 能夠不增加快取記憶體存取時間，並同時達到減少電源消耗之目地。

此計畫執行第一年，我們選擇 MPEG-2 軟體解碼程式作為第一個使用這種省電管理的應用程式。我們實驗的結果在快取記憶體的耗電量方面可以達到 40%左右的省電量，同時不影響程式執行的效能。

關鍵詞：能量消耗，快取記憶體，電源延遲最佳化

Abstract

Power consumption is becoming a critical design issue because portable devices (e.g., hand-held computing and personal telecommunication devices) increase in popularity. Cache memories account for a significant fraction of a chip's overall energy dissipation. In this project, we propose an informed cache architecture that utilizes application-specific information for energy • delay optimization. One commonly used technique to save power on a cache access is to enable smaller cache structures. However, reducing power often comes at the cost of sacrificing arbitrary amounts of performance because of not being able to predict where requested data exist in cache memories accurately. Informed cache architecture employs a hardware-software cooperative scheme that assigns different types of data in a program to specified regions of cache memories. By explicitly controlling the cache resource allocation, we can avoid increasing cache access latency while reducing cache energy dissipationPower consumption is an important design issue of current embedded systems. Data caches consume a significant portion of total

processor power for data intensive applications. In this project, we propose to utilize application-specific information for cache resource allocation to achieve energy saving, including cache bypassing, the mini-cache and way-partition.

We use a software MPEG-2 video decoder as our first targeted application to test the effectiveness of the proposed mechanism. The results show up to 40% of cache energy reduction without sacrificing performance.

Keywords: Power consumption, cache memories, energy • delay optimization

二、Introduction & Objective

Power consumption is becoming a critical design issue of embedded system due to the popularity of portable devices such as cellular phones and personal digital assistants. It has been reported that caches consume a significant portion of the total processor power. For example, 42% of processor power is dissipated in the cache subsystem in StrongARM 110 [1]. Many embedded applications, in both the multimedia and communication domains, are data dominated. Data storage and transfer account for a significant portion of overall power consumption. Whether a reference goes to the main memory or not, it must access the data cache. Therefore, techniques to reduce energy dissipation in the data cache are critical to deliver an energy-efficient embedded system.

Cache partitioning and way-prediction are two commonly used techniques to reduce energy dissipation in data caches. Cache partitioning schemes divide caches into smaller components since a smaller cache has a lower load capacitance. Way-prediction predicts the matching way and probes only the matching way instead of all ways to reduce power consumption for set-associative caches. These techniques often increase average access latency if the referenced data is not located in the predicted region.

The need for prediction is due to the fact that cache management is transparent to software. If we allow software to control cache resource allocation, we can access the region where a memory reference is located directly. In this way, we can achieve energy saving without increasing average cache access time. Allowing software to control caches has been proposed to improve cache performance for embedded systems [2][3]. In this paper, we exploit the potential of using a software-managed cache for energy optimization.

We use a software MPEG-2 video decoder as our first targeted application. An MPEG-2 decoder has large data set and requires high data processing rate, which are two important characteristics of real-time signal processing applications. We consider three software-controlled cache management mechanisms and demonstrate how to utilize the application-specific information of an MPEG-2 decoder to achieve energy saving. Cache bypassing saves energy by accessing the L2 cache directly for data that have little reuse. The mini-cache scheme stores frequently accessed data with small memory footprints into a small on-chip memory area. Way-partition maps program data structures to different ways of set-associative caches according to their working set size and access frequency. On each access, we can access the matching

ways directly instead of probing all ways as in the traditional cache design.

We can break down the data types in a MPEG-2 decoder into the following classes:

Input— The MPEG-2 bitstreams.
Output— The decoded picture data.
Tabular— Static and read-only tables used in the decoder.
Reference— Buffers for both current and reference frames.
Block—The buffer for pixel values of a single macroblock.
State—Variables needed for setting and operation of the decoder.

Table 1 lists the data set size and percentage of total memory references for different data types. Note that the access percentage from the major data types only adds up to 82%. A significant portion of the remaining references comes from accessing the stack region (12% of total memory references). Based on this information, we can determine cache allocation policy for individual data type. The experimental results show up to 40% energy savings from the proposed software-managed cache mechanism. We have submitted the paper to ICICS-PCM 2003. Below we summarize the results.

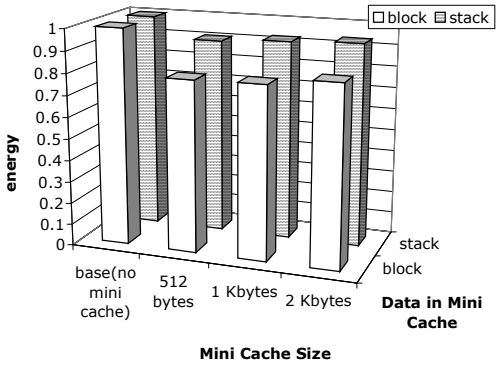


Figure 1. The mini-cache energy consumption

ıta type	siz
ɔut	2K
tput	500K
ɔular	5K

able 1: Summar
emory references

三、Results Summary

Since energy-saving methods may reduce energy dissipation at the expense of performance degradation, we evaluate the performance, energy consumption and energy-delay product of the proposed energy-saving techniques. The proposed energy-saving techniques may increase L1 cache miss rate, thereby increasing the L2 cache energy consumption. Therefore, for a fair comparison, we consider both the L1 and L2 caches for energy evaluation.

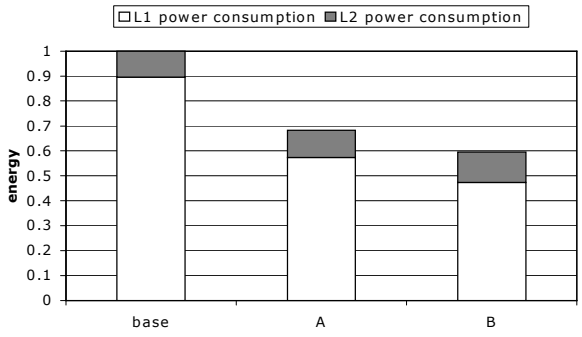
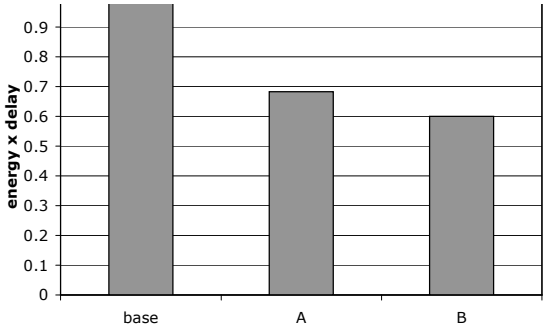


Figure 2: Normalized energy consumption of way-partition.

A: block (1 way);others (3 ways)
B: block+state (1 way); tabular+stack (1 way);others (2 ways)

Cache Bypassing

Based on the attributes of different data types listed in Table 1, the video output data of the MPEG-2 decoder is an ideal data type for bypassing since the output stream is written and never read by the CPU. The experimental results show that excluding video output data from the L1 cache can reduce the energy consumption by 1.4%. It also offers slight performance improvement. We do not see significant energy saving because the output data accounts for only 2% of total memory references.



Normalized energy-delay

Figure 3: The effect of way-partition.
A: block (1 way);others (3 ways)
B: block+state (1 way); tabular+stack (1 way);others (2 ways)

The mini-cache

Here we evaluate the energy efficiency of storing the block data type of an MPEG-2 decoder to the mini-cache. Several studies propose to have a separate partition for the stack data references [7][14]. Mapping the block data type to the mini-cache achieves higher energy saving compared to the stack memory references (21% vs. 10%). On the performance side, the addition of a mini-cache only offers slight performance improvement because the baseline model has already very low L1 cache miss rate (1.02%).

Way-Partition

The cache resource allocation strategy used in the way-partition mechanism is to give frequently accessed data types priority and allocate resources close to their working set sizes[1]. We consider two partitioning schemes. The first scheme reserves one way of the L1 cache for the block data type and maps other data types to the remaining three ways. The second scheme aggressively partitions data into three groups: block+state (1 way), tabular+stack (1 way) and others (2 ways). A finer partitioning saves more energy of the L1 cache since each access consumes less power but it could cause more capacity and conflict misses. Therefore, the tradeoff between performance and energy saving needs to be carefully evaluated.

Figure 2 shows the normalized energy consumption of these two partitioning schemes. We divide the energy consumption into the L1 and L2 components. The results show significant energy saving. The first scheme reduces the energy consumption by 31.8% and the second by 40.4%. Both partitioning schemes have only little effect on the L2 power consumption. That implies insignificant performance impact. The normalized energy-delay product is shown in Figure 3. The second partitioning scheme reduces the energy-delay product by 40%.

四、Conclusion

In this project, we propose to use a software-managed cache for energy optimization for a software MPEG-2 video decoder. We evaluate three energy-reduction techniques. This study has shown the potential of using a software-management cache for energy reduction. In future work, we plan to investigate compiler techniques for automatic cache resource allocation.

五、 Acknowledge

Students who join this project are 曾宏偉 楊善詠, 辛逸軒, and 李建豪.

六、Bibliography

[1] J. Montanaro, et al. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE Journal of Solid State Circuits*, 31(11):1703-1714, November 1996

[2] D. Chiou, P. Jain, L. Rudolph and S. Devadas. Application-Specific Memory

[1] We define working set size as the smallest cache size required to obtain a specific miss ratio.

Management for Embedded Systems Using Software-Controlled Cache. In *Proceedings of DAC*, 2000. Los Angeles, California

[3] P. Soderquist and M. Leeser. Memory Traffic and Data Cache Behavior of an MPEG-2 Software Decoder. In *Proceedings of International Conference on Computer Design*, 1997

[4] T. L. Johnson, D. A. Connors, M. C. Merten, and W. W. Hwu. Run-Time Cache Bypassing. *IEEE Transactions on Computers*, Vol. 48, No. 12, December 1999, pp. 1338-1354

[5] B. Case. SPARC V9 Adds Wealth of New Features. *Microprocessor Report*, 7 (9), February 1993

[6] J. Kin, M. Gupta, W. H. Mangione-Simith. The Filter Cache: An Energy Efficient Memory Structure. In *Proceedings of 30th Annual International Symposium on Microarchitecture*, December, 1997

[7] H.-H. Lee and G. S. Tyson. Region-Based Caching: An Energy-Delay Efficient Memory Architecture for Embedded Processors. In *Proceedings of International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2000)*, Nov. 2000.

[8] O. S. Unsal, I. Koren, C. M. Krishna and C. A. Mortiz. The Minimax Cache: An Energy-Efficent Framework for Media Processors. In *Proceedings of 8th International Conference on High Performance Computer*, Febuary 2002

[9] M. D. Powell, A. Agarwal, T. N. Vijaykumar, B. Falsafi and K. Roy. Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping. In *Proceedings of 34th Intel Symposium on Microarchitecture*, 2001

[10] C.-L. Su and A. Despain. Cache Design Tradeoffs for Power and Performance Optimization: A Case Study. In *Proceedings of International Symposium on Low Power Design*, Apr. 1995, pp. 63-68

[11] David H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. *Journal of Instruction-Level Parallelism*, 2000

[12] S.-H Yang, M. D. Powell, B. Falsafi, and T. N. Vijaykumar. Exploiting Choices in Resizable Cache Design to Optimize Deep-Submicron Processor Energy-Delay. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, November 2001

[13] S.-H Yang, M. D. Powell, B. Falsafi, K. Roy, and T .N. Vijaykumar. An Integrated Circuit/Architecture approach to reducing leakage in deep-submicron high-performance I-cache. In *Proceedings of the 7th IEEE Symposium on High-Performance Computer Architecture*, Jan 2001.

[14] M. Huang, R. Reanu and J. Torellas. L1 Cache Decomposition for Energy Efficient Processors. In Proceedings of Internatinal Symposium on Low-Power Electronics and Design (ISPLED'01), Huntington Beach, CA, August 2001.

[15] P. R. Panda, N. D. Dutt and A. Nicolau. Efficient Utilization of Scratch-Pad Memory in Embedded Processor Applications. In *Proceeding of European Design & Test Conference*, 1997

[16] Intel StrongARM SA-1110 Microprocessor Brief Datasheet, April 2000

[17] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and optimizations. In *Proceedings of the*

27th International Symposium on Computer Architecture (ISCA), Vancouver, British Columbia, June 2000.

[18] S. Echart and C. Fogg. ISO/IEC MPEG-2 Software Video Codec. In *Proceeding of the SPIE conference on Digital Video Compression: Algorithms and Technologies*, Vol. 2419, 7-10 February 1995, San Jose, California, pp. 100-109.

°