行政院國家科學委員會專題研究計畫　成果報告

嵌入系統中低功率快取記憶體結構之設計(2/2)

計畫類別： 個別型計畫
計畫編號： NSC92-2213-E-002-014-
執行期間： 92 年 08 月 01 日至 93 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

計畫參與人員： 曾宏偉　喻秉鴻　陳依蓉　施澤聰

報告類型： 完整報告
報告附件： 出席國際會議研究心得報告及發表論文
處理方式： 本計畫可公開查詢

中　華　民　國 93 年 12 月 15 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　※
※　　嵌入系統中低功率怪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　93 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 喻秉鴻 陳依蓉 施澤聰

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：台灣大學資訓工程學系

中　華　民　國　93　年 10　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 93 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，我们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 dissipation Power 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

Data type	size	Access%	Data type	size	Access%
input	2KB	0.2%	Reference	1500KB	27.5%
output	500KB	1.9%	block	1.5KB	34.2%
tabular	5KB	9.2%	State	1.5KB	9.1%

Table 1: Summary of decoder data types, size and % of memory references

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

Data type	(Associativity,size)
Input	(1, 32 Bytes)
Output	(1, 32 Bytes)
Tabular	(1, 2 KBytes)
Reference	(2, 256 Bytes)
Block	(1, 1 KBytes)
State	(2, 1 KBytes)
Stack	(1, 512 Bytes)
Others	(1, 512 Bytes)

Table 2: Required minimal cache resources for each data type

data type should be allocated as few ways as possible. Therefore, we test cache configurations in the increasing order of the degree of set-associativity. For each possible size given an associativity, we calculate the number of cache misses from the tested data type. If the number of misses is not greater than that in the base architecture, the minimal resources for that data type is identified. No more profiling runs are needed for that data type. The proposed algorithm is listed below, and the profiled result is shown as in Table 2.

```
Algorithm 1. : Profiling Methodology
1:  for each data type do
2:    for set associativity, $S$(1 to n),where n is the degree of associativity of L1 do
3:      for cache size, $T$ (in power of 2, $\leq S \times size\_per\_way$) do
4:        calculate $no\_misses$
5:        if $no\_misses \leq no\_base\_misses$ then
6:          report($S$,$T$)
7:          goto the next data type
8:        end if
9:      end for
10:   end for
11: end for
```

The second phase is to determine the mapping between data types and cache regions based on the profiling information. We first identify data types that are eligible for the mini-cache; that is, the required associativity is one and the size is not greater than the mini-cache size. To achieve more energy savings, we would like to have more memory accesses satisfied in the mini-cache. Therefore, we place the most frequently accessed data types to the mini-cache provided that their size is not greater than the mini-cache size. The proposed mini-cache selection algorithm is shown as the following:

```
Algorithm 2. : Mini-cache Selection
1: Select data types with $S = 1$ and $T \leq mini\_cache\_size$,
   where (S,T) represents the minimal required cache resources (associativity, size)
2: Sort the selected data types in the no-decreasing order of their access frequencies
3: The first $k$ types of the sorted list are mapped to the mini-cache, where $size$(the first $k$ data types) $\leq$ $size$(mini-cache) and $size$(the first $k+1$ data types) $>$ $size$(mini-cache)
```

The last step is to determine the mapping between ways of the L1 cache and the rest of data types (referred to as way-partition). The allocation principle is to keep the utilization of each cache way as even as possible. In other words, a data type should be mapped to ways with minimal utilized resources. We quantify the utilized resources of a way as the total sizes of data types allocated to that way. The mapping information for each data type is represented as a bit vector. For example, if a data type is mapped to the first 2 ways of a 4-way cache, its mapping vector is (1,1,0,0). The details of the way-partition algorithm is summarized below.

```
Algorithm 3. : Way-partition
1: Keep a set of ways in a sorted list in the non-decreasing order of $\sqcup$, where $\sqcup$ represents the amount of utilized resources
2: for each data type $D$ with required cache resources ($S$,$T$) do
3:   for first $S$ ways of the way list do
4:     $\sqcup = \sqcup + \frac{T}{S}$
5:   end for
6:   Derive the mapping vector
7:   Sort the way list in the non-decreasing order of $\sqcup$
8: end for
```

The paper has been accepted by IEEE Transactions on Circuits and Systems for Video Technology. Below we summarize the results.

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.

Software controlled cache

Before presenting the effectiveness of the software-controlled cache, we first summarize the minimal resources required for each data type in Table 2. The minimal resources for data types are gathered by our proposed profiling algorithm.

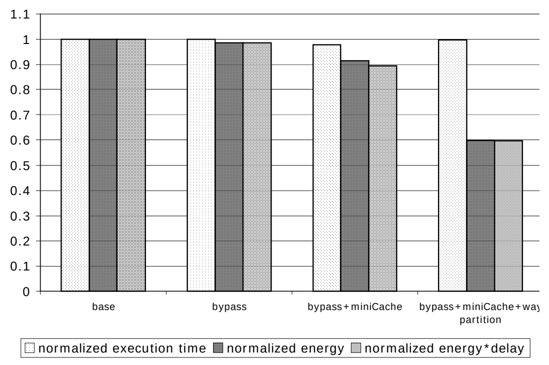

Figure 1: The normalized execution time, energy consumption, and energy-delay product of different software-controlled techniques

We can see that the required cache size is smaller than the data set size presented in Table 1. For example, the data set size of the reference data type is 1500 KB while the required cache size is only 256Bytes. For data types that are not allocated contiguously in memory, such as reference and state, multiple ways are required to eliminate self-interferences. Based on our proposed mini-cache selection algorithm we proposed:

the stack data are mapped to the mini-cache. The way allocation information of the rest data types are decided by our proposed way-partition algorithm. Note that the output data bypasses the on-chip caches.

To analyze the effectiveness of the proposed software-controlled cache in details, we evaluate the proposed scheme in three configurations:bypassing,bypassing+mini-cache, bypassing+mini-cache+way-partition. Figure 3 shows the execution time, energy and ED normalized to the base architecture for these three configurations. We can see that bypassing output data reduces the energy by 1.4%. The performance advantage of bypassing is not significant. For the MPEG-2 software decoder, the output data only accounts for 2% of memory references. For applications performing output operations more often, we expect the bypassing mechanism should present more performance and energy advantages. Mapping the stack data to the mini-cache in addition to bypassing can achieve up to 9.5% energy reduction with slight performance improvement. Partitioning the L1 cache among the remaining data types can further reduce the energy by more than 30% without performance degradation. In summary, the proposed software-controlled cache achieves up to 40% energy reduction while maintaining the comparable performance as the base architecture.

To further demonstrate the importance of our proposed data allocation policy. We compare the above data allocation with another one derived simply by the information presented in Table 1. The output data still bypass the on-chip caches, and the stack variables are mapped to the mini-cache since it is the only data type whose size is not greater than the mini-cache. The rest of data types are mapped to different ways of the L1 cache. Note that the allocation policy still tries to keep the utilization of each way as even as possible. Since the differences between these two data allocations are in how the L1 cache is partitioned among data types, we focus on the comparison between these two way-partition methods. A good way-partition method should reduce the energy consumed in the L1 cache without increasing that of other memory components; that is, way-partition should not increase L1 misses significantly. The optimized-partition by our proposed data allocation policy consumes 28% less L1 energy than that of coarse-partition with less than 1% differences in the L1 miss ratios. Note that the slight increase in the L1 miss ratios comes from the interferences among data types allocated to the same region.

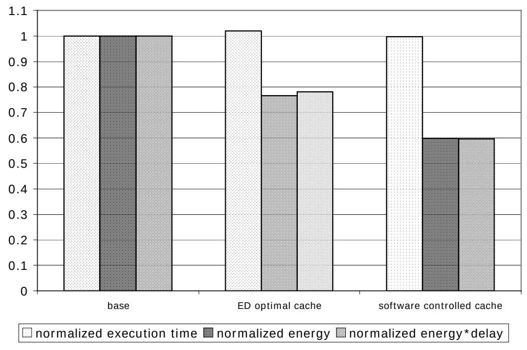

Figure 2: The normalized execution time, energy consumption, and energy-delay product of best design space point and algorithm optimization

Software-Controlled Cache v.s. ED Optimal Cache

To find the ED optimal cache architecture for the software MPEG-2 decoder, we perform a thorough design space exploration. We vary the cache size from 4K to 128K, degree of set associativity from direct-mapped to 4-way and block size from 8 to 128B. A 64B, 2-way 8K cache has the lowest ED value among the configurations tested.

Figure 2 compares the normalized execution time, energy and ED of the software-controlled cache and the ED optimal cache (64B, 2-way, 8K). All three metrics are normalized to the base architecture (32B, 4-way, 8K). We can see that the ED optimal cache performs slightly worse than the base architecture but it consumes 23% less energy. This tells us that if we simply use the cache architecture in an ARM-like processor, a significant amount of cache energy is wasted. However, by employing the proposed software-controlled cache technique, an ARM-like cache architecture consumes even less energy than a dedicated cache tuned for the MPEG-2 software decoder.

四、Conclusion

In this project, we propose to use a software-controlled cache for energy-efficiency optimization on a shared cache architecture of an integrated multimedia system. The proposed software-controlled cache allocates data types in an application to different cache regions. A data type is either mapped to the mini-cache, ways of the L1 cache or bypass on-chip caches. The optimization goal is to achieve energy reduction without performance degradation. We test the effectiveness of the software-controlled cache on the MPEG-2 software decoder. The results show that the software-controlled cache reduces 40% of energy on an ARM-like cache architecture without sacrificing performance. It consumes even less energy in comparison with a cache architecture tuned specifically for the MPEG-2 decoder. This study has shown that the proposed software-controlled cache does provide a way to improve the energy efficiency of a shared cache architecture for an integrated multimedia system on an application-specific basis. In the future, we will apply the software-controlled cache technique to other multimedia applications, such as the MPEG-4 decoder.

This research work has been published in the IEEE Transactions on Circuits and System for Video Technology, special issue on integrated multimedia system.

五、 Acknowledge

Students who join this project are 曾宏偉, 喻秉鴻 陳依蓉 施澤聰.

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 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.

°