Circuits and Systems, 2014, 5, 253-264
Published Online November 2014 in SciRes. http://www.scirp.org/journal/cs
http://dx.doi.org/10.4236/cs.2014.511027

Scientific Research

Reducing Power and Energy Consumption of Nonvolatile Microcontrollers with Transparent On-Chip Instruction Cache

Dahoo Kim, Itaru Hida, Eric Shun Fukuda, Tetsuya Asai, Masato Motomura

Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
Email: kim@lalsie.ist.hokudai.ac.jp, hida@lalsie.ist.hokudai.ac.jp, fukuda@lalsie.ist.hokudai.ac.jp, asai@ist.hokudai.ac.jp, motomura@ist.hokudai.ac.jp

Received 25 August 2014; revised 20 September 2014; accepted 6 October 2014

Abstract

Demands for low-energy microcontrollers have been increasing in recent years. Since most microcontrollers achieve user programmability by integrating nonvolatile (NV) memories such as flash memories for storing their programs, the large power consumption required in accessing an NV memory has become a major problem. This problem becomes critical when the power supply voltage of NV microcontrollers is decreased. We can solve this problem by introducing an instruction cache, thus reducing the access frequency of the NV memory. Unlike general-purpose microprocessors, microcontrollers used for real-time applications in embedded systems must accurately calculate program execution time prior to its execution. Therefore, we introduce a "transparent" instruction cache, which does not change the existing NV microcontroller's cycle-level execution time, for reducing power and energy consumption, but not for improving the processing speed. We have conducted detailed microar chitecture design based on the architecture of a major industrial microcontroller, and we evaluated power and energy consumption for several benchmark programs. Our evaluation shows that the proposed instruction cache can successfully reduce energy consumption in a fairly wide range of practical NV microcontroller configurations.

Keywords

Embedded System, Microcontroller, Instruction Cache, Nonvolatile, Low-Power Design

1. Introduction

In recent years, sensor networks have been widely studied as a fundamental technology to help realize the concept of the "**SmartSociety**" [1]. In order to implement sensor networks in various fields of application, sensor

How to cite this paper: Kim, D., Hida, I., Fukuda, E.S., Asai, T. and Motomura, M. (2014) Reducing Power and Energy Consumption of Nonvolatile Microcontrollers with Transparent On-Chip Instruction Cache. *Circuits and Systems*, **5**, 253-364.
http://dx.doi.org/10.4236/cs.2014.511027

nodes that can operate for a long time with a small energy source are required. Therefore, it is necessary to reduce power consumption of microcontrollers that operate sensor nodes.

Meanwhile, NV microcontrollers (microcontrollers integrated with on-chipnonvolatile memories) are widely used due to their convenience in helping develop embedded system software. However, the power consumption of the nonvolatile memory dominates the total power consumption of the microcontroller [2]. Furthermore, it is difficult to reduce the power consumption of nonvolatile memories in microcontrollers. The purpose of our work is to reduce power and energy consumption by introducing an instruction cache to the microcontroller, which will reduce the frequency of access of the nonvolatile memory.

As shown in **Table 1**, traditional cache architecture research has attempted to improve microprocessor performance by introducing a high-speed cache memory, known as static random-access memory (SRAM), between the main memory and the datapath by reducing memory access time [3]. On the other hand, in the case of NV microcontrollers used for real-time applications in embedded systems, the program execution time can be calculated in advance. Furthermore, the change of execution due to cache misses should be avoided, since such a change can cause problems in the system such as real-time applications. Thus, it is necessary to introduce an instruction cache that does not cause cache miss penalties and leaves the speed of memory access during cache hits unchanged.

Therefore, in this paper, we attempt to reduce power and energy consumption instead of improving performance. To accomplish this, we introduce a "transparent" instruction cache that does not change the cycle-level timing of existing NV microcontrollers. In particular, we evaluate power and energy reduction by introducing the transparent instruction cache to a base microcontroller that is integrated with a flash memory, which is the most popular nonvolatile memory.

The rest of this paper is organized as follows: Section 2 describes the features of our research. Section 3 describes the architecture of the proposed instruction cache employed with the NV microcontroller. Section 4 discusses the evaluation results of the instruction cache's effects of reducing power and energy consumption. Section 5 concludes this paper.

2. Features of Our Research

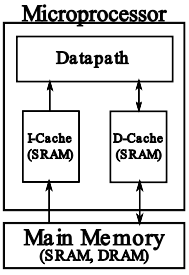
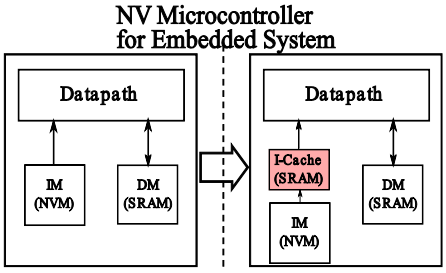
The features of our research are as follows:

1) Examination of instruction cache suitable for microcontroller deployments: As described in Section 1, in the case of a microcontroller, it is more important to prolong the battery run time than to reduce the processing time. Thus, the instruction cache that we propose in this paper is intended to reduce power consumption rather than to improve processing speed.

2) Evaluation based on a realistic microcontroller architecture: We evaluated the power and energy consumption of our system, which was built on a base microcontroller, Renesas Electronics Corporation's 78K0R. Since not all of the 78K0R's specifications are open, we implemented the base microcontroller using only publicly available information [4]-[6].

3) Power evaluation with high precision: We evaluated the power and energy consumption using the chip design data generated from the RTL description. This allowed us to obtain the results with high precision.

Table 1. Concept of this work.

Traditional Cache Architecture Research	This work
	
High Performance	Low power and low energy transparent (*i.e.*, doesn't change cycle level timing)

3. Architecture

3.1. Base Microcontroller

Our microcontroller is based on the 78K0R, which is used in various industrial fields [7]. The block diagram is shown as a part of **Figure 1**. The 78K0R has an on-chip flash memory as its NV instruction memory, and a SRAM as its data memory [4].

This architecture has a three-stage pipeline structure (IF stage, ID stage, MEM stage) [5]. In the IF stage, the microcontroller provides an address from the program counter (PC) to the instruction memory (flash memory), and fetches an instruction sequence from the instruction memory. This instruction sequence is stored in an instruction queue (I-Queue). In the ID stage, the microcontroller decodes the instruction that has been fetched in the IF stage, and extracts data memory (SRAM) and register file (RF) addresses to be accessed. In the MEM stage, the microcontroller retrieves the data from the data memory and executes the instruction.

In addition, the base microcontroller uses complex instruction set computing (CISC) architecture [5]. The number of instructions of the base microcontroller is 915, and the instruction length is 1 byte to 5 bytes. The base microcontroller has a 4-byte (1 word) instruction queue that contains an instruction sequence fetched from the instruction memory. Therefore, if a valid instruction exists in the I-Queue, there is no need to access the flash memory.

3.2. 1-Word-Per-Line Instruction Cache

Most (on-chip) NV microcontrollers, including the 78K0R, access the NV memory in one cycle [5]. Therefore, ourbase microcontroller fetches one word from the instruction sequence in the flash memory in one cycle [5]. Thus, as the first step to introduce the instruction cache to the base microcontroller, we designed a 1-word-per-line instruction cache architecture, as shown in **Figure 1**. We assumed the operation timing of the instruction cache to be the same as the timing of the base microcontroller's access to the flash memory: The instruction sequence is read from flash memory in the subsequent cycle of the access to the flash memory.

Operation of the instruction cache is as follows:

1) In the case of a cache miss, the microcontroller accesses the flash memory and fetches 1 word from the instruction sequence.

2) In the case of a cache hit, the microcontroller accesses the instruction cache (I-Cache) and fetches 1 word from the flash memory. The word is written to the instruction cache. Since the bit widths of the instruction cache and the flash memory are the same, there is no penalty for writing to the instruction cache.

Since the proposed instruction cache does not allow for a cache miss penalty, there is a problem in that a read access and a write access to the instruction cache can occur coincidentally. For example, as shown in **Figure 2**,

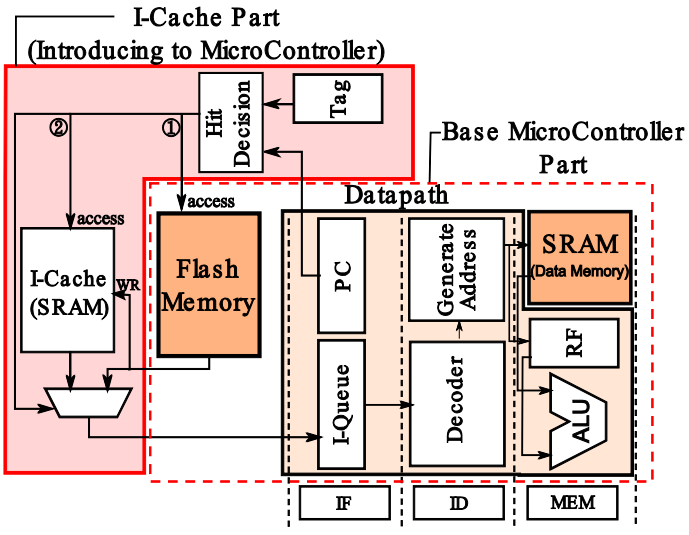


Figure 1. Concept of proposed 1-word-per-line instruction cache architecture.

when a cache miss occurs at PC1, the microcontroller reads the flash memory. Then, in the next cycle, the instruction sequence is read from the flash memory. Thus, the instruction sequence for the cache miss (F-DATA1) must be written to the instruction cache in this cycle (the next cycle of the cache miss). In the cycle immediately after the cache miss, when a cache hit occurs at PC2, an instruction is fetched from the instruction cache. In this case, a read access and a write access collide in the instruction cache.

There are two methods for solving this problem:

1) Clock cycle dividing method: On collision of read access and write access in the instruction cache, this method bisects the clock cycle and allows write access in the first half and read access in the second half.

2) Intermediate buffer insertion method: This method delays the write access to the cache by inserting an intermediate buffer to which the instruction sequence in the cache miss is written between the instruction cache and the flash memory. **Figure 3** shows its architecture. In this method, the timing of writing the instruction sequence to the instruction cache is a cycle that does not access the instruction cache. If a valid instruction is still in the I-Queue of the datapath as described in the base microcontroller architecture, and if a cache miss occurred, the microcontroller does not have access to the instruction cache.

In particular, when a cache miss occurs, the instruction sequence that has been present in the intermediate buffer is written to the instruction cache, and the instruction sequence that has been read from the flash memory is newly written to the intermediate buffer.

In the case of the clock cycle dividing method, controlling the divided clock cycle is complicated. In addition, because it is necessary to generate a write/read access signal in one cycle, the operating frequency of the microcontroller must be lowered unless another clock is provided. Therefore, in this paper, we adopt the intermediate buffer insertion method, which simplifies the control scheme.

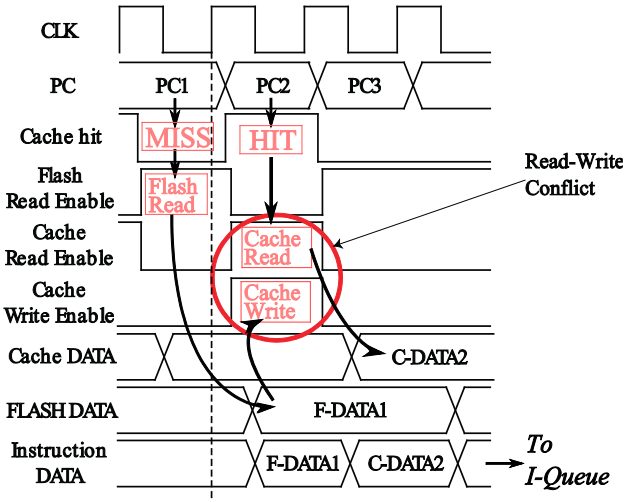


Figure 2. Instruction memory access timing in case of cache hit immediately after cache miss.

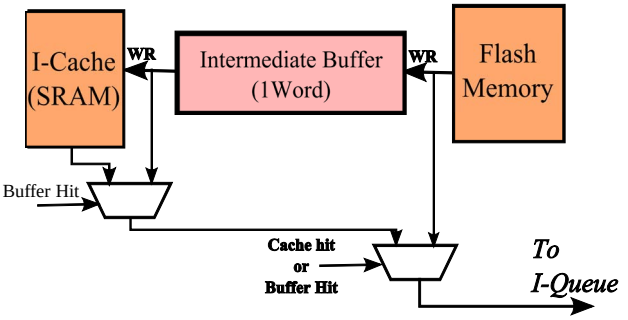


Figure 3. Intermediate buffer insertion method for 1-word-per-line instruction cache architecture.

3.3. 4-Word-Per-Line Instruction Cache

For the second step to integrate the instruction cache to the base microcontroller, we designed a 4-word-per-line instruction cache architecture that can take advantage of spatial locality.

The flash memory (NV memory) of the base microcontroller (the existing NV microcontroller) reads one word from the instruction sequence in one cycle. Therefore, in order to implement the 4-word-per-line instruction architecture, a buffer for storing four words is required. We implemented a 4-word-per-line instruction cache by extending the number of words in the intermediate buffer, as shown in **Figure 4**. The instruction sequence that is read from the flash memory in one cycle is stored in the intermediate buffer at one word per cycle. When all four entries of the intermediate buffer are filled, the contents are sent to the 4-word-per-line instruction cache memory.

For example, when a miss occurs in the first cycle, the instruction sequence that is read from the flash memory is stored to one of the entries of the intermediate buffer by referring to the lower 2 bits of the instruction address (held in the PC). In the next cycle, if the flash memory is not accessed (in other words, if there are cache hits, buffer hits, or I-Queue hits), the next instruction sequence is read from the flash memory by incrementing the PC, and is stored to the next entry of the intermediate buffer. Four words of the instruction sequence will be written to the instruction cache memory.

However, when a miss occurs before the four words are collected in the intermediate buffer, it is necessary to store the instruction sequence that has been read from the flash memory to the intermediate buffer. In this case, the existing instruction sequences in the intermediate buffer are written to the instruction cache, and the instruction sequence that has been read from the flash memory on a miss is stored to the intermediate buffer. For example, as shown in **Figure 5**, if a miss occurs when only three words of an instruction sequence are stored in the intermediate buffer, the three words are written to the instruction cache, invalidating the 0th word. In this paper, we extend the 4-word-per-line tag memory's valid bits to 4 bits, and make them indicate the valid word of each line.

However, when writing the line that has invalid words to the instruction cache, such as the case shown in **Figure 5**, it is expected that there will be a case of a low hit rate, due to the deletion of the valid word that existed in one line of the instruction cache. In Section 4, we will evaluate our system in this regard.

3.4. Associativity

As the third step to integrate the instruction cache with the base microcontroller, we increased the associativity from 1 to 2 and 4. It is expected that the power consumption of the control unit of the tag memory will increase

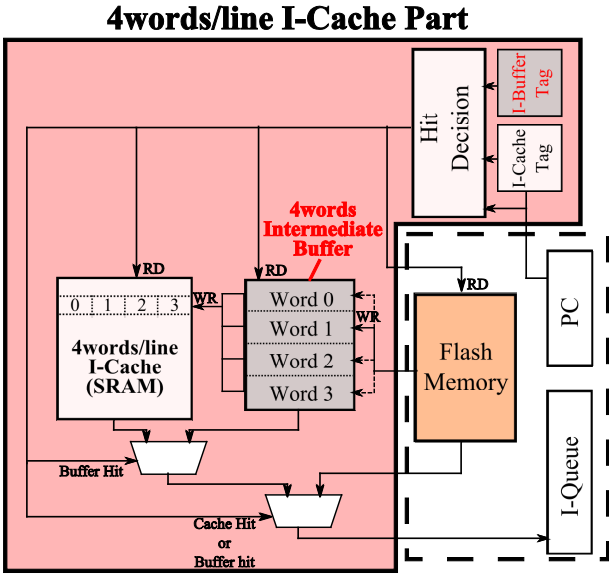


Figure 4. Proposed 4-word-per-line instruction cache architecture.

because the control unit of the tag memory becomes complicated by increasing the associativity. Yet there is a possibility that the hit rate will rise. We will evaluate our system in this regard as well. In addition, we adopt a pseudo least-recently-used (LRU) replacement algorithm.

4. Evaluation

4.1. Method of Evaluation

Figure 6 shows our evaluation method. In the first step, we made a RTL description of the base microcontroller based on the public specifications of the 78K0R, and attached the instruction cache architecture to the RTL description. In the second step, we generated the chip design data of the base microcontroller with and without the instruction cache. The chip design data was generated through a logic synthesis and a placement/routing, using the RTL description. The tools used for the logic synthesis and the placement/routing are shown in **Table 2**. In the third step, we evaluated the power and energy consumption of each instruction cache system by examining the results of a back-annotated simulation, using the chip design data generated in the second step.

We used the evaluation programs shown in **Table 3**. The method of estimating power consumption is as follows:

1) CPU logic: We estimated the power consumption from the results of the back-annotated simulation using chip design data with the Synopsys Power Compiler.

2) Data memory: We evaluated the power consumption of the base microcontroller's data memory (SRAM) using the SRAM's power consumption parameters, which are publicly available [8].

3) Instruction cache memory: Similarly, we evaluated the power consumption from publicly available information about the SRAM's power consumption parameters. Further, we changed the size of the cache memory from 128 bytes to 4 Kbytes and evaluated it.

4) Tag memory: Tag memory was also evaluated using the SRAM information.

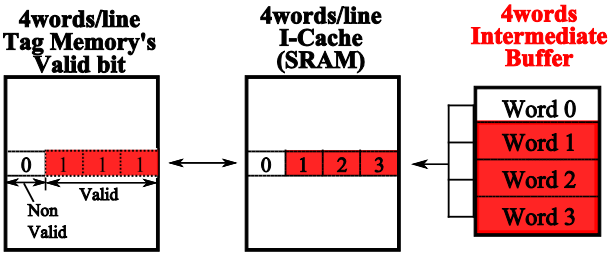


Figure 5. Writing operation of proposed 4-word-per-line instruction cache.

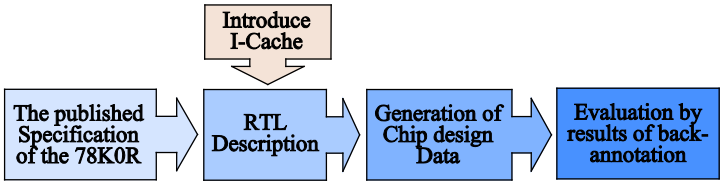


Figure 6. Method of evaluation for this work.

Table 2. Tools and process used in evaluation.

Phase	Tools and Process
Logic Synthesis	Synopsys Design Compiler (ver. 2013.03-SP3)
Placement and Routing	Cadence Encounter (ver. 10.13)
Estimated Power Consumption	Synopsys Power Compiler (ver. 2013.03-SP3)
Process	TSMC 0.18 μm

Table 3. Evaluation programs and hit rate.

Evaluation Program	Size (Byte)	1 Word/Line (%)	4 Word/Line (%)
Bubble Sort	614	98.69	98.76
Celsius to Fahrenheit	585	96.76	96.11
Checksum	535	96.92	98.46
Copy Verify	602	97.90	98.18
Factorial	606	98.63	95.67
EEMBC Coremark	11701	99.89	93.03

In addition, we used a 1.8 V power supply voltage and a 20 MHz operating frequency for evaluating power and energy consumption. Operation frequency was determined from the data arrival time of the critical path of its architecture, after logic synthesis. Furthermore, we evaluated energy consumption using the power consumption and delay time of the instruction cache, before and after its introduction.

4.2. Evaluation of Flash Memory (NV Memory)

In general, it is known that the power consumption of the flash memory access occupies a large part of the entire power consumption of a flash (NV) microcontroller. Moreover, the power consumption of flash memory differs according to the memory's capacity, the device technology, and circuit configurations. In addition, power consumption specifications of the NV memory are not publicly available.

In this paper, we evaluated the power consumption of the flash memory by parameterizing the ratio that the power consumption of flash memory occupies (*i.e.*, 30%, 50%, and 70%). We selected these parameters to indicate the energy reduction in a wide range of practical NV microcontroller configurations. Thus, by realizing reduced power and energy consumption of the proposed instruction cache, we believe we can provide useful information to microcontroller designers.

4.3. Power Consumption

First, the hit rates for the evaluation programs are shown in **Table 3**. The cache size is 2 Kbytes. Because the programs are small in size, they (from the bubble sort program to the factorial program) showed high hit rates. Hereafter, we will evaluate power consumption in greater detail using the EEMBC Coremark programin order to discuss power consumption more realistically.

In the EEMBC Coremark program, the hit rate of each cache size is shown in **Figure 7**. For a large instruction cache (1 Kbytes or more), the 1-word-per-line instruction cache has a higher hit rate than the 4-word-per-line instruction cache. This is because there is also a case where valid data that was in the instruction cache in the 4-word-per-line instruction cache has been discarded as described in the 4-word-per-line instruction cache architecture. In the case of an instruction cache of less than 1 Kbyte, the 4-word-per-line instruction cache has a higher hit rate. This is because the 4-word-per-line instruction cache can take advantage of spatial locality, even if the valid data has been discarded. Further, if the associativity was high, the hit rate was high for both methods. In particular, when the cache size was small, this effect became significant.

Next, we evaluated actual power consumption. Estimated power consumption results for an instruction cache of 512 bytes are shown in **Figure 8**. The results show that the proposed instruction cache reduces the power consumption of the flash memory by 66% with a 1-way, 1-word-per-line instruction cache, and by 90% with a 1-way, 4-word-per-line instruction cache. The larger effect on the 4-word-per-line instruction cache is a result of its higher hit rate.

In addition, for the 4-way instruction cache, the reduction of power consumption increased by 4% for the 1-word-per-line instruction cache, and by 2% for the 4-word-per-line instruction cache, compared to a 1-way instruction cache. However, the power consumption of the instruction cache memory and the tag memory shown in **Table 4** is added after introducing the instruction cache.

For the instruction cache memory, since the bit width of the 4-word-per-line instruction cache is larger than that of the 1-word-per-line instruction cache, the power consumption for accessing the 4-word-per-line instruction

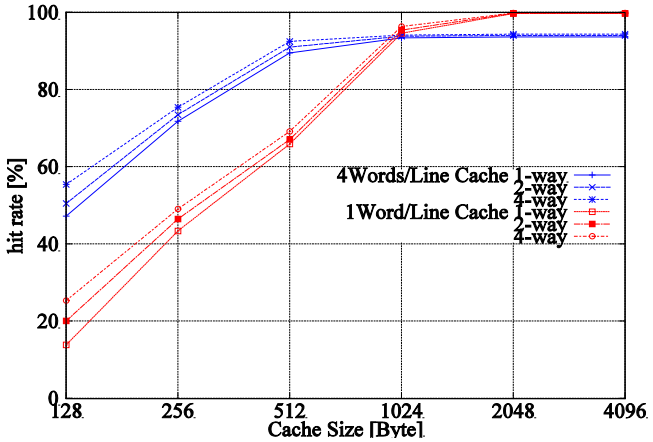


Figure 7. Cache hit rate by cache size. (Evaluation program: EEMBC Coremark)

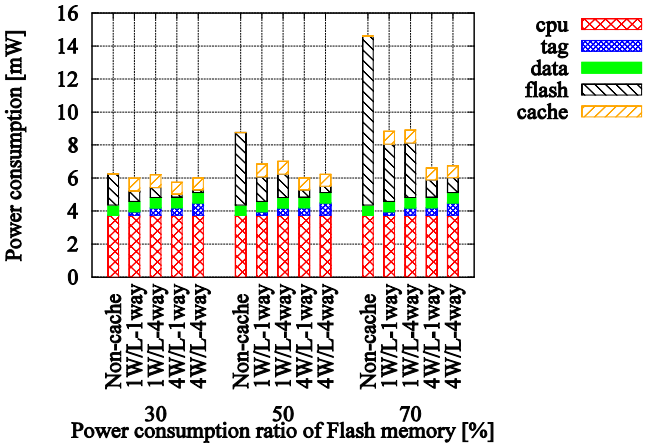


Figure 8. Power consumption by ratio of flash memory's power consumption to overall power consumption of microcontroller. (Cache size: 512 bytes)

Table 4. Power consumption of instruction cache and tag memory. (Cache size: 512 bytes)

Evaluation Program	Cache (mW)	1 Way Tag (mW)	4 Way Tag (mW)
1 Word/Line Instruction Cache	0.775	0.209	0.449
4 Words/Line Instruction Cache	0.775	0.461	0.749

cache memory does not increase significantly. This occurs because the architecture often has cache hits at the intermediate buffer, and the access rate to the cache memory is lower than that of the 1-word-per-line cache.

For the tag memory, power consumption increases as the width of the instruction cache line and the associativity increase, since the control circuit becomes complicated when the bit width of the tag memory increases.

From the above results, we note that the absolute amount of reduced power consumption by the proposed instruction cache becomes larger as the ratio of the flash memory's power consumption increases. In addition, when their capacities are 512 bytes, the 4-word-per-line instruction cache has a larger effect on reducing power consumption than the 1-word-per-line instruction cache. As associativity increases, power consumption decreases for the 4-way instruction cache.

Next, we evaluated power consumption by changing the size of the instruction cache. The results are shown in **Figure 9**. The red line in this figure represents power consumption without the instruction cache. There is a case where total power consumption with the instruction cache increases when the size of the 1-word-per-line in-

struction cache is 128 bytes (1-way associative). This is because the reduced power consumption of the flash memory is small (because the hit rate for this cache size is low), and the power consumption of the instruction cache memory and tag memory has been added.

In addition, reflecting the behavior of the hit rate, the 4-word-per-line instruction cache is more effective at reducing power consumption for a small instruction cache (512 bytes or less). Similarly, in the case of a large (1 Kbyte or more) instruction cache, the 1-word-per-line instruction cache is more effective in reducing power consumption. In addition, as associativity increases, the reduction of power consumption becomes larger as the hit rate improves in the case of a small (128-byte) instruction cache. In other cases, the power consumption does not decrease as much because the power consumption of the tag memory increases.

Furthermore, we investigated the crossover point, which is the point where power consumption before and after the introduction of the instruction cache are the same. If the ratio of the power consumption of the flash memory exceeds the crossover point, power consumption can be expected to be reduced by introducing the instruction cache. The evaluation results of the crossover point are shown in **Figure 10**.

In the case of a 512 byte or larger instruction cache, the crossover point was approximately 20% to 30%. Accordingly, if the power consumption of the flash memory was approximately 20% or more, a reduction in power consumption can be expected by introducing a 512 byte or larger instruction cache. In the case of a 256 byte or smaller instruction cache, a reduction in power consumption cannot be expected in the 1-word-per-line instruction cache. However, in this case, the reduction in power consumption improved slightly by introducing a 4-way associative instruction cache. However, for other sizes of instruction cache, the crossover point increased because power consumption did not decrease as much.

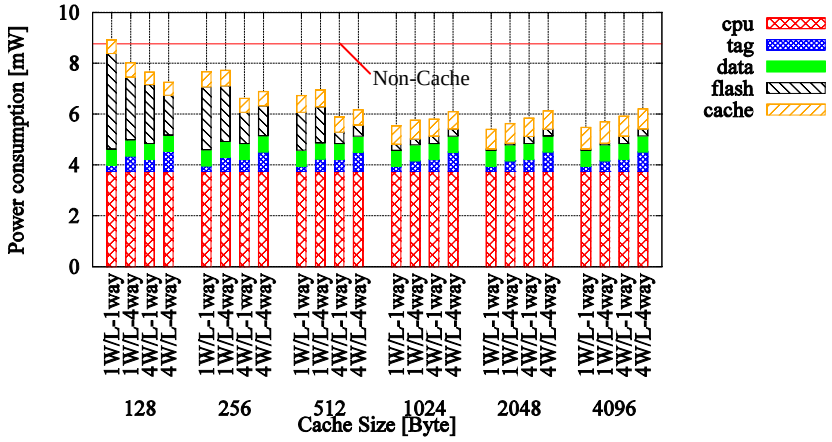


Figure 9. Power consumption by cache size. (Ratio of flash memory's power consumption: 50%)

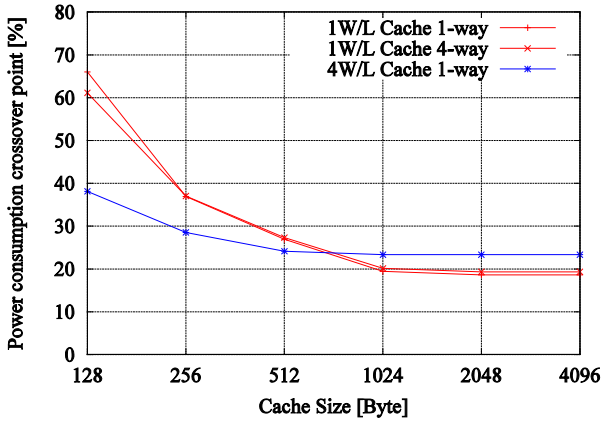


Figure 10. Change of crossover point by cache size for normal voltage operation.

4.4. Reducing the Power Supply Voltage

In recent years, research in reducing the power consumption of logic circuits by lowering the power supply voltage has been reported [9]. However, lowering the power supply voltage of flash memories or other NV memories is difficult, owing to the circuit characteristics of NV memories [10]. Because the proposed instruction cache can work at a lower power supply voltage in accordance with the logic circuits, and because it can decrease the frequency of yet high-power flash memory access, the power reduction should be more enhanced.

Thus, we examined the effect of reducing power consumption by introducing a proposed instruction cache with a decreased power supply voltage (to 0.9 V) for all components except for the flash memory. In addition, we used a 4 MHz operating frequency, which was estimated from a circuit simulation of a single gate.

Calculating with the following Formula (1), the ratio of the flash memory's power consumption at normal voltage is shown in **Table 5**. By decreasing the power supply voltage, the power consumption of the flash memory becomes dominant, compared to the power consumption during normal voltage operation.

$$P \propto C_L V_{dd}^{\ 2} f_{clk} \tag{1}$$

The crossover points during low-voltage operation are shown in **Figure 11**. With a lower voltage, the crossover point is reduced by approximately 20% from the crossover point during normal voltage operation. This is because the absolute amount of power consumption by the proposed instruction cache decreases more as the power consumption of the flash memory becomes dominant. Therefore, during low-voltage operation, a reduction in power consumption can be expected for an even smaller ratio of the flash memory's power consumption. When the crossover point in the instruction cache of 512 bytes or more becomes approximately 10%, a reduction in power consumption can be expected by introducing an instruction cache of 512 bytes or more, even if the power consumption of the flash memory occupies approximately 10% of the entire power consumption at normal voltage.

4.5. Area and Critical Path

Using the proposed instruction cache, various increased areas of the microcontroller chip are shown in **Table 6**. These values indicate an area of the proposed instruction cache control logic and cache memory (SRAM). We estimated the area using the following method:

- Proposed instruction cache control logic: We calculated this area based on the chip design data of the post placement and routing.
- Cache memory (SRAM): We calculated this area using memory design data that was obtained by using MEMAKER by Faraday Technology.

The critical path of the NV microcontroller is generally a read-path of NV memory. This is because NV memory is slower than that of the other logic types [11]. In addition, operation frequency is determined by the critical path

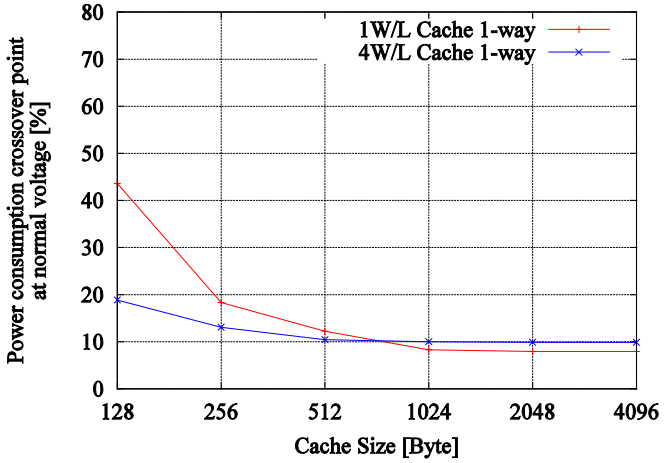


Figure 11. Change of crossover point by cache size for low-voltage operation.

(read-path of NV memory). By introducing the proposed instruction cache, the delay time of the logic path increases. This increase occurs because of the cache hit determination of the proposed instruction cache. In most cases, the increased delay time of the cache hit determination will not exceed the existing critical path of the NV microcontroller (read-path of NV memory). In most cases, the delay time of the NV memory read-path is higher than that of the logic path [11]. However, if the increased delay time of the logic path by cache hit determination exceeds the existing critical path, the delay time of the critical path will increase up to 2.91 ns for a 1-word-per-line instruction cache, and 3.62 ns for a 4-word-per-line instruction cache. These delay times are calculated using the chip design data of the post placement and routing.

4.6. Low-Energy Effect

In this section, we will show the low-power-consumption effects and the low-energy-consumption effects by examining the reduction rate of power consumption for the ratio of the power consumption of the flash memory, which is 30% to 70%.

First, we determine the low-power-consumption effects for 512 byte and 2 Kbyte instruction caches, as shown in **Table 7**. In the case of the 512 byte instruction cache, the effect of the 4-word-per-line instruction cache is larger than that of the 1-word-per-line instruction cache, and the obtained low-power-consumption effect is approximately 16.7% to 56.5%. For the 2 Kbyte instruction cache, the effect of the 1-word-per-line instruction cache is larger, and the obtained low-power-consumption effect is approximately 37.5% to 75.6%. In addition, a low-power effect was confirmed for low-voltage operation.

In the worst case, the delay time due to hit determination was approximately 2.91 ns for the 1-word-per-line instruction cache, and approximately 3.62 ns for the 4-word-per-line instruction cache. We determined the low-energy-consumption effects on the basis of these results, as shown in **Table 8**. Reflecting the delay time of the hit determination, the low-energy-consumption effect decreased by approximately 5% for the 1-word-per-line instruction cache, and approximately 6% for the 4-word-per-line instruction cache.

Table 5. Change of ratio of power consumption of flash memory.

Normal Voltage Operation	Low-Voltage Operation
30%	63%
50%	80%
70%	90%

Table 6. Area of proposed instruction cache.

Cache Size (Byte)	1-Word-per-Line Instruction Cache	4-Word-per-Line Instruction Cache
256	0.112, mm^2	0.241, mm^2
512	0.133, mm^2	0.265, mm^2
1024	0.167, mm^2	0.301, mm^2
2048	0.220, mm^2	0.369, mm^2

Table 7. Low-power-consumption effect.

	Cache Size: 512 Bytes (4 Words/Line)	Cache Size: 2 Kbytes (1 Word/Line)
Normal Voltage Operation	Approximately 16.7% - 56.5%	Approximately 37.5% - 75.6%
Low Voltage Operation	Approximately 23.1% - 64.3%	Approximately 47.4% - 84.1%

Table 8. Low-energy-consumption effect.

	Cache size: 512 Bytes (4 Words/Line)	Cache size: 2 Kbytes (1 Word/Line)
Normal Voltage Operation	Approximately 10.1% - 52.4%	Approximately 33.3% - 74.4%
Low Voltage Operation	Approximately 16.7% - 60.0%	Approximately 44.4% - 83.1%

5. Conclusions

In this paper, we proposed a transparent instruction cache architecture for reducing the power and energy consumption of a practical NV microcontroller architecture. Unlike traditional cache architecture, we intended to reduce power and energy consumption rather than improve processing speed. This is because, in the case of a NV microcontroller used for real-time applications in embedded systems, it is important to prolong the battery run time and not change the existing NV microcontroller's cycle-level execution time.

In addition, we presented the effects of reducing power and energy consumption by introducing proposed instruction caches. The proposed instruction cache reduces energy consumption by 10% to 52% for a ratio of the power consumption of the flash memory of 30% to 70%, if the size of the cache is one-twentieth of the program size. Meanwhile, for NV memories that include flash memories, it is generally difficult to lower the power supply voltage due to the circuit characteristics. Therefore, the proposed instruction cache is more effective in reducing energy consumption by lowering the power supply voltage, since the proposed instruction cache can work at a lower power supply voltage and can decrease the frequency of NV memory accesses. In this case, the proposed instruction cache reduces energy consumption by 16%, to 60%.

From the above evaluation results, we can see that the proposed instruction cache can successfully reduce the energy involved. In particular, if the power consumption of the flash memory occupied approximately 20% or more of normal voltage operation, a reduction in power consumption can be expected by introducing a 1 Kbyte or smaller instruction cache. In addition, microcontroller designers can lower a logic circuit's power supply voltage in order to reduce energy consumption. In this case, if the power consumption of flash memory is approximately 10% or more, a reduction in power consumption can be expected by introducing a 512-byteor smaller instruction cache.

Our future work involves devising a architecture for increasing the hit rate of the instruction cache, and to evaluate the plurality of a realistic large-size application. In addition, we intend to evaluate more precise power consumption using a prototype chip.

References

[1] Gungor, V.C., Lu, B. and Hancke, G.P. (2010) Opportunities and Challenges of Wireless Sensor Networks in Smart Grid. *IEEE Transactions on Industrial Electronics*, **57**, 3557-3564. http://dx.doi.org/10.1109/TIE.2009.2039455

[2] Lee, H.G. and Chang, N. (2003) Energy-Aware Memory Allocation in Heterogeneous Nonvolatile Memory Systems, *Proceedings of the* 2003 *International Symposium on Low Power Electronics and Design*, 25-27 August 2003, 420-423. http://dx.doi.org/10.1109/LPE.2003.1231941

[3] Goodman, J.R. (1983) Using Cache Memory to Reduce Processor-Memory Traffic. *Proceedings of the* 10*th Annual International Symposium on Computer Architecture*, June 1983, 255-262. http://dx.doi.org/10.1145/285930.285984

[4] Renesas Electronics Corporation, 78L0R/KE3 User's Manual. http://documentation.renesas.com/doc/DocumentServer/U17854EJ9V0UD00.pdf

[5] Renesas Electronics Sorporation 78K0R Microcontroller User's Manual: Instructions. http://documentation.renesas.com/doc/DocumentServer/r01us0029ej0600_78k0r.pdf

[6] Renesas Electronics Corporation, 78L0R/Kx3-A User's Manual: Hardware. http://documentation.renesas.com/doc/DocumentServer/r01uh0003ej0200_78k0rkx3a.pdf

[7] Oba, K., Kawai, K., Matsushita, R., Ishihara, K. and Eto, K. (2009) Development of 16-Bit All Flash Microcomputers "78K0R/Kx3-L" Featuring Ultralow Power Consumption. *NEC Technical Journal*, **4**, 35-39. http://www.nec.com/en/global/techrep/journal/g09/n01/pdf/090109.pdf

[8] Katevenis, M. (2003) On-Chip SRAM. http://www.csd.uoc.gr/~hy534/03a/s31_ram_bl.htm

[9] Chandrakasan, A.P. and Brodersen, R.W. (1995) Minimizing Power Consumption in Digital CMOS Circuits. *Proceedings of the IEEE*, **83**, 498-523. http://dx.doi.org/10.1109/5.371964

[10] Bez, R., Camerlenghi, E., Modelli, A. and Visconti, A. (2003) Introduction to Flash Memory. *Proceeding of the IEEE*, **91**, 489-502. http://dx.doi.org/10.1109/JPROC.2003.811702

[11] Park, C., Seo, J., Seo, D., Kim, S. and Kim, B. (2003) Cost-Efficient Memory Architecture Design of NAND Flash Memory Embedded Systems. *Proceedings of the* 21*st International Conference on Computer Design*, 13-15 October 2003, 474-480. http://dx.doi.org/10.1109/ICCD.2003.1240943

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.