Memory Organization for Low-Energy Processor-Based Application-Specific Systems

Cao, Yun
Department of Computer Science and Communication Engineering, Kyushu University

Yasuura, Hiroto
Department of Computer Science and Communication Engineering, Kyushu University

出版情報：IEICE Transactions on Electron. J83-C (8), pp.1616-1624, 2002-08. Institute of Electronics, Information and Communication Engineers
バージョン：
権利関係：

PAPER

Memory Organization for Low-Energy Processor-Based Application-Specific Systems

Yun Cao[†] *and* **Hiroto Yasuura**[††]**,** *Members*

SUMMARY This paper presents a novel low-energy memory design technique based on variable analysis for on-chip data memory (RAM) in application-specific systems, which called VAbM technique. It targets the exploitation of both data locality and effective data width of variables to reduce energy consumed by data transfer and storage. Variables with higher access frequency and smaller effective data width are assigned into a smaller low-energy memory with fewer bit lines and word lines, placed closer the processor. Under constraints of the number of memory banks, VAbM technique use variable analysis results to perform allocating and assigning on-chip RAM into multiple banks, which have different size with different number of word lines and different number of bit lines tailored to each application requirements. Experimental results with several real embedded applications demonstrate significant energy reduction up to 64.8% over monolithic memory, and 27.7% compared to memory designed by memory banking technique.

key words: Low Energy, Memory Organization, Variable Analysis, Application-Specific System

1. Introduction

Memory-processor integration on System-on-Chips offers new opportunities for reducing the energy of embedded systems. One of the key issues in the design of energy-efficient processor-based architectures for embedded systems is the power consumed by memories [1]. Several researchers have pointed out that the power consumption in memories can take a dominant fraction on the power budget of a whole embedded system for data-dominated applications [2] [3]. Embedded processor-based systems allow for customization of on-chip memory configuration based on application-specific requirements [4]. Memory size including not only the number of word lines, but also the number of bit lines can be tailored to application requirements. Therefore, application-specific memory architectures can be developed to minimize energy consumption for a given embedded application.

Several researchers have analyzed the power dissipation of different memory architectures for a given application. Paper[5] presented power reduction technique for instruction memories in application specific systems. Paper[6] presented a methodology for developing models of on-chip SRAM memory organization. Paper[7] proposed a power-minimization approach to simultaneous register and memory allocation in behavior synthesis. The allocation problem is formulated as a minimum-cost network flow that can be solved in polynomial time. The rationale behind this approach is to map variables that have nonoverlapping lifetimes.

On-chip caches are well known architectural optimization technique for memory design [8]. Our work moves from the observation that cache memories are not the most power-efficient architecture. Information storage/retrieval into/from a cache is much more power-consuming than accessing a memory containing the same amount of data. In embedded systems, on-chip SRAM (scratch-pad SRAM) is a valid alternative to caches. In this architecture, the most frequently accessed addresses are statically mapped onto scratch-pad SRAM to guarantee power and performance efficiency. The main difference between the scratch-pad SRAM and data cache is that the SRAM guarantees a single-cycle access-time, whereas access to the cache is subject to cache misses.

The problem of efficiently utilizing memory banks in DSPs was addressed. Techniques for partitioning variables for simultaneous access into two memory banks of the Motorola 56000 DSP processor are reported in [9] and [10]. Paper[11] [12] focused on-chip SRAM bank partitioning for low energy consumption. They started from the dynamic execution profile of an embedded application running on a given processor core, and synthesize a multi-banked SRAM architecture optimally fitted to the execution profile.

As far as we known, this paper first focuses on memory organization (allocation and assignment) for low-energy on-chip memory with respect to the memory utilization considering not only memory access frequency but also effective data width of variables. We first present an exploration technique for determining efficient on-chip memory architecture, characterized by the size of Scratch-pad memory with different number of bit lines and word lines, based on variable analysis for a given application, targeting to reduce energy consumption of memory.

In application-specific design, because the entire application is to some extent statically known, and this knowledge let us perform memory allocation and assignment more intelligently to reduce the memory energy consumption. We generate an application-specific hierarchical memory architecture that exploits the lo-

Manuscript received October 3, 2001.
Manuscript revised February 8, 2002.

cality, which caused by non-uniform access to memory. The more important, we allocate memory also considering effective data width of variables, because in many applications, there are a lot bits in variables, which are never used during program execution. The rationale of our memory organization technique is to assign those variables with higher access frequency and smaller effective data width into a smaller low-energy application-specific memory with fewer numbers of bit lines and word lines, which is placed closer the processor. Our experiments show that the exploration results can provide critical feedback to the designer about the optimal memory configuration for a given application. This paper focuses only on the data memory organization because for many embedded applications, the volume of data being processed far exceeds the number of instructions.

The rest of this paper is organized as follows. Section 2 gives preliminaries. Section 3 describes variable analysis. Section 4 presents our technique for low-energy memory organization. Experiments and results are reported in section 5. Finally, Section 6 concludes and gives our future work.

2. Preliminaries

This paper focuses on a customized memory organization for low energy on-chip memory in data-dominated applications. The power consumption related to memory transfer and storage dominates overall system power [3]. Hence, we perform memory allocation and assignment considering both memory access frequency and effective data width of variables to explore low-energy memory organization.

2.1 Motivation

The task of memory energy optimization for application-specific systems is to create low-energy memory architectures customized for a given application. Low levels of hierarchy are small, fast, and energy efficient memories, while high levels of hierarchy are relatively large, slow, and energy hungry memories. The motivation for hierarchical memory organization is exploitation of temporal and spatial storage locality to reduce the total energy dissipation of the memories.

Energy consumption strongly depends on the size of physical memory (both of the number of bit lines and word lines). Memories with smaller size consumed lower energy. Therefore, we try to allocate variables with higher access frequency and smaller effective data width into a smaller memory with fewer numbers of bit lines and word lines, which leads significant energy reduction. However, allocating too many memory banks leads increase of energy consumption because of addressing complexity and severe wiring overheads. For a given embedded application program, our goal is to determine the memory allocation and assignment by mapping each variable into on-chip Scratch-Pad SRAM, while minimizing the application's overall memory energy consumption.

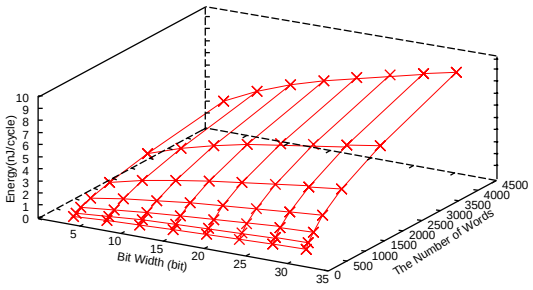

Fig. 1 Bit width and word count vs. energy consumption of read access for SRAM

2.2 Size-Energy Correlation of Memory

Because a static random access memory(SRAM) does not require additional fabrication steps and dedicated technology, it can be easily integrated onto the same chip with the processor and other logic circuits. Therefore, embedded SRAMs are much more common in SoC design than non-volatile memories and DRAMs. In this work, we focus on on-chip SRAMs, because it will probably become mainstream in the next few years [13].

For memories, we assume that the power dissipated for charging the global bit line is in proportional to the number of partitioned segments, and the power dissipated in a single segment is in proportional to the size of the segment. Under these assumptions, the memory power consumption can be approximated by formulation(1), where N_{seg}, N_{word}, η, λ, and γ denote the number of segments, the number of words, and coefficients for each term, respectively.

$$E_{mem} = \eta \cdot N_{seg} + \lambda \cdot \frac{N_{word}}{N_{seg}} + \gamma \quad (1)$$

In formulation(1), the first and second terms represent the energy dissipated for charging the global bit line and the energy dissipated in a single memory segment respectively. The last term represents a constant factor in memory power consumption. From formulation(1), it is easy to derive that the number of memory segments which minimizes the memory power consumption is $\sqrt{(\lambda/\eta) \cdot N_{word}}$. We generated some SRAM models by Alliance CAD System $Ver.4.0$ with $0.5um$ double metal CMOS technology, and using the SPICE simulation of these memories with the different configurations, we obtained the estimation models of SRAM as follows:

$$e_r = 24.9 \cdot \sqrt{b \cdot N_{word}} + 56[pJ/cycle] \quad (2)$$

$$e_w = 197 \cdot \sqrt{b \cdot N_{word}} + 369[pJ/cycle] \quad (3)$$

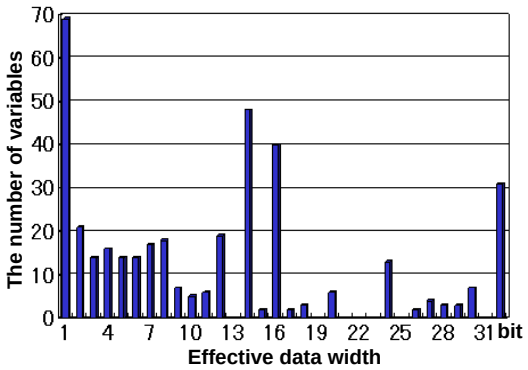

Fig. 2 Variable distribution for effective data width in MPEG-2 decoder

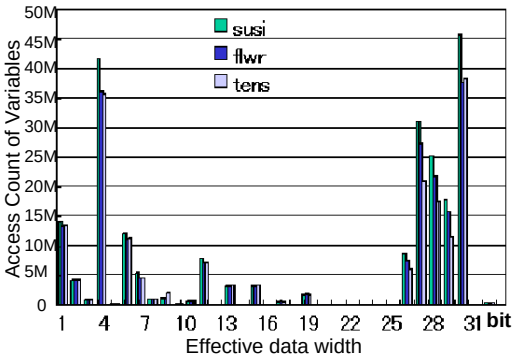

Fig. 3 Access count of variables for MPEG-2 decoder

Where The access energy of memories for read/write operations is expressed as e_r and e_w respectively. b is the bit width of memory and N_{word} is the number of words.

Figure1 shows the bit width and the number of words VS. energy consumption of read access for our SRAM modules. It demonstrates that the energy consumption in SRAMs strongly depends on the bit width and the number of words. The energy consumption of SRAM reduces with the decrease of the number of bit lines and word lines.

3. Variable Analysis

In application-specific design, to some extent, the entire application is statically known. Therefore, by analyzing effective data width of variables we can reduce unused bits to reduce memory storage and more over by analyzing the lifetime of variables, we can make efficient use of memory storage. In addition, we reduce memory access energy by analyzing variable access frequency.

3.1 Data Width Analysis

In many cases, there are a lot of bits of a variable are never used during execution of a program. Therefore, the effective data width of each variable in an application program needs to be analyzed in order to use memory efficiently [14], which results in reduction of energy consumption.

This section describes methods to analyze effective data width of variables in C programs, because C language is familiar for most designers. In this paper, we define *effective data width* as the smallest size which can hold both maximum and minimum values of a variable [15] [16]. If a variable x of unsigned integer type whose value is in [0, 2000], then the number of necessary bits of x is 11, because the 11-bit size is large enough to hold any value in [0, 2000]. We use two methods to analyze effective data width of variables, one is static analysis, and the other is dynamic analysis.

For static analysis, when the maximum value of an unsigned integer variable x is n_{max}, the effective data width of x, $e(x)$, is given as follows:

$$e(x) = \log_2(n_{max} + 1) \tag{4}$$

For a signed integer x with a maximum value n_{max} and a minimum value n_{min}, $e(x)$ is defined as follows:

$$e(x) = \lceil \log_2 \mathcal{N} \rceil + 1 \tag{5}$$

where

$$\mathcal{N} = \max(|n_{max}| + 1, |n_{min}|) \tag{6}$$

Static analysis is an efficient method to analyze the effective data width of variables. However, in many cases when we can not predict the assigned value of a variable unless we execute the program, such as the case of unbounded loops, static analysis becomes insufficient. As a solution to this problem, we adopted dynamic analysis. In dynamic analysis, we execute the program and monitor the values assigned to each variable, and then analyze the required data width of the variable.

We analyzed the C source program of MPEG-2 video decoder using our developed variable analyzer and got the variable analysis results of effective data width depicted in figure2. This figure shows that there are a lot of variables having many unused bits in MPEG-2 decoder, which originally declared as “int” type.

3.2 Lifetime Analysis

The lifetime of a variable, defined as the period between its definition and last use [17]. It is an important metric affecting register allocation, where variables with disjoint lifetimes can be stored in the same register. Just like this, we analyze lifetime of variables [18], so that we can cluster them to share same memory. Therefore, by analyzing lifetime of variables, we can make efficient use of memory to achieve energy reduction.

3.3 Access Frequency Analysis

It is well known that only a few parts of programs are frequently executed in many application programs. Therefore, to profile the access frequencies of the variables in theses programs and assign them into a small memory is a effective way for low energy design. The main purpose of this step is to find a memory organization with good storage locality for frequently accessed memory locations. We have built a profiler, and use it by simulation, we got the figure3 shown the profiled results for the access frequencies of variables with effective data width in MPEG-2 decoder. From the figure, we can see that there are "hot spots" for variable access in MPEG-2 decoder.

4. Memory Organization for Low Energy

Memory organization can be divided into two steps. First, several memories are chosen from the available memory modules with different number of bit lines and word lines. This is called memory allocation; second, the variables are assigned into these allocated memories. The step is called memory assignment. When variables are assigned into memories, the size of variables determines the required memory size and the maximal data width determines the required bit width of memories. With the decision of memory allocation and assignment, the memory organization is fully determined.

Our target is to find an optimal allocation and assignment for low-energy application-specific memory. The target system assumed as the follows: Data memories are organized by several on-chip SRAMs having different size with different number of bit lines and word lines, a hierarchical memory.

The memory model is shown in Figure 4. Various memory allocation and assignment can be seen as specializations of the structure. In the hierarchical model, low hierarchy levels are made of small memories with few numbers of bit lines and word lines, close to processor, and tightly coupled with it. Memories at high hierarchy levels are made of increasingly size with increasing number of bit lines and word lines, far from processor. Roughly speaking, the distance between a processor and a memory hierarchy level represents the effort needed to fetch (or store) a given amount of data from (to) the memory. Effort can be expressed in units of energy. On the other hand, memory levels increases addressing complexity and has a sizable area overhead. Both these factors reduce the power savings.

As a concrete example of a memory architecture that can be modeled with the template of Figure 4, the hierarchy has three levels. The first level N1 has 16 bytes lines with 8 bit width, the second level N2 has 64 bytes with 16 bit width, and the third level N3 has 128 bytes with 32 bit width. Similarly, memory size increases with level. Average energy consumption per

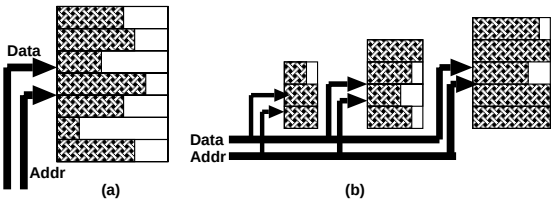

Fig. 4 (a) Monolithic memory (b) Allocated and assigned memory using our technique

cycle for read accessing a N1 memory is approximately 337pJ; for N2 memory access is 852pJ; and for N3 access is 1132pJ.

Our memory optimization technique is based on the following facts:

- The smaller memory size becomes, the less energy will be consumed. We consider not only the number of words but also the bit width of memories.
- Access to memory are highly non-uniform.
- There are a lot of variables, whose many bits are never used during program executions.
- Variables with disjoint lifetime can share same memory space.

4.1 Our Approach

The main purpose of memory allocation and assignment is to minimize overall energy cost within performance and memory size constraints. Hierarchical organizations derived from the generic template of Figure4 reduce memory energy consumption by exploiting not only the nonuniformities in access frequencies but also effective data width of variables. We generate a hierarchy memory, whose frequently accessed locations are placed in low hierarchy levels with small size (few numbers of bit lines and word lines).

The procedure of our memory allocation and assignment approach for low energy consists of the following phases:

- Phase 1: For a given application program, analyze variables, report effective data width and access frequency of variables.
- Phase 2: Formulate energy dissipation of SRAM modules as the function of memory size including bit width and the number of words. In this paper we use formulation (2) and (3) as the SRAM energy models.
- Phase 3: Use analysis results including lifetime, access frequency and effective data width of variables, perform memory allocation and assignment considering not only the number of words but also the bitwidth of memory, in order to minimize memory energy consumption.

Detail will be explained in section 4.4, VAbM technique.

4.2 Energy Cost Metrics

The energy cost function employed for estimating memory energy is shown as following. Memory energy dissipation per cycle requires energy models. We adopt the model shown in section 2, formulation (2) and (3) are empirically derived from simulation and, unlike other analytical models, they are expressed in terms of high-level parameters, the number of words and bit width. The total energy consumption of memories is the summation of the memory banks. For each memory bank, we estimate its energy consumption for read and write respectively. thin space

$$TEm = \sum_{N}(TEm(j) + TEm_{\delta}(j)) \tag{7}$$

$$TEm(j) = TEm_r(j) + TEm_w(j) \tag{8}$$

$$TEm_r(j) = \sum_{x_i \in Q_j} e_r(x_i) \times TNa_r(x_i) \tag{9}$$

$$TEm_w(j) = \sum_{x_i \in Q_j} e_w(x_i) \times TNa_w(x_i) \tag{10}$$

$$TEm_{\delta}(j) = Emon \times \delta(j) \tag{11}$$

where

TEm : Total energy consumption of memory
N : Total number of memory banks
$TEm(j)$: Energy consumption of memory bank j
$TEm_{\delta}(j)$: Energy overhead for added bank j
$Emon$: Energy consumption of monolithic memory
$TEm_{r/w}(j)$: Energy consumption of memory bank j for read(write) access
X : A finite set of variables in a given application program, $X = \{x_1, x_2, ..., x_n\}$
x_i : A variable, $x_i \in X$
$e_r(x_i)(e_w(x_i))$: Energy consumption per read (write) access for variable x_i
$TNa_r(x_i)(TNa_w(x_i))$: The number of read (write) accesses for variable x_i
$Q(j)$: the subset of variables assigned into bank j $Q(j) \subseteqq X$
$\delta(j)$: Overhead coefficient for added bank j, caused by addressing complexity.

4.3 Problem Formulation

This section gives some assumptions and notations, and then formulates the problem of low-energy memory allocation and assignment.

Assumption

- We assume that we can generate arbitrary size of SRAM. The size of the Scratch-Pad SRAM is limited by the total area available on-chip.
- We assume that register allocation, which assigns frequently accessed variables such as loop indices to processor registers, has already been performed.
- In order to determine the optimal memory allocation and assignment for a given global variable, we should observe its references over all procedures, but normally a compiler generates code one procedure at a time. For parameters are similar when they are passed as pointers. This problem can be addressed by using inter-procedural data-flow analysis. In this paper, we assume that all global variables and parameters can be statically assigned into on-chip memory.
- During the address register allocation phase, arrays and other aggregate variables become complicated, in this paper we simply treat each array element as a distinct variable.

Notation

- $X = \{x_1, x_2, ..., x_n\}$: A finite set of variables
- $EWd = \{EWd(x_1), EWd(x_2), ..., EWd(x_n)\}$: A set of effective data width for variables x_i
- $TNa = \{TNa(x_1), TNa(x_2), ..., TNa(x_n)\}$: A set of total memory access for variables x_i
- N_{Max} : The maximum number of allowed memory banks
- $\delta(j)$, TEm, N, $TEm(j)$, $TEm_{\delta}(j)$, $Q(j)$: Defined in previous section.

Problem Formulation

Given X, EWd, TNa and N_{Max}
To find N and $Q(j)$
So that $TEm = \sum_N(TEm(j) + TEm_{\delta}(j))$ is minimized
Subject to
$N \leqq N_{Max}$, $Q(j) \subseteqq X$
$X = Q(1) \bigcup Q(2) ... \bigcup Q(N)$
$Q(1) \bigcap Q(2) ... \bigcap Q(N) = \phi$

Allocated and assigned memory

N banks, each bank has $b(j) \times m(j)$ size (the number of bit lines and word lines are $b(j)$, $m(j)$ respectively)

where

$b(j) = \max_{x_i \in Q(j)} EWd(x_i)$, $m(j) = |Q(j)|$

Given all these definition, our memory allocation and assignment approach will be given in detail in our VAbM technique.

4.4 VAbM Technique

This section presents VAbM technique for application-specific memory organization in detail. VAbM receives a given application program AP as input, and generates the customized memory architecture having N banks along with the assignment of the variable subset $Q(j)$ into each memory bank with size of $b(j) \times m(j)$ bits as output, which have optimized energy consumption. Each variable will be served by only one local memory module, determined statically.

The goal of VAbM technique is that under the constraints of the maximum number of memory banks

Input:
source program : AP
(variables : $x_i \in X_{org} = \{x_1, x_2, ..., x_n\}$, $1 \leqq i \leqq n$)
input data : D_{in}
the maximum number of banks : N_{Max}
overhead coefficient : $\delta(j)$ $(1 \leqq j \leqq N)$

Variable:
the number of memory banks $N \leqq N_{Max}$
the subset of variables $Q(j) \subseteqq X$

Output:
the optimized number of banks N
the optimized subset of variables $Q(j)$
the optimized energy consumption TEm_{opt}
the optimized memory with size $b(j) \times m(j)(bits)$

Step 1 : Variable Analysis
{
$analyzer(AP, D_{in})$;
$LT = \{LT(x_1), LT(x_2), ..., LT(x_n)\}$,
$TNa = \{TNa(x_1), TNa(x_2), ..., TNa(x_n)\}$,
$return(EWd = \{EWd(x_1), EWd(x_2), ..., EWd(x_n)\})$;
}

Step 2 : Memory Allocation and Assignment

$X \leftarrow CompactVar(X_{org}, LT)$;
$X_s \leftarrow SortVar1(X, LT)$;
$I(j) \leftarrow InitPart(X_s, N_{Max})$;
$P(j) \leftarrow GetopP(I(j), LT)$;
$C(j) \leftarrow SortVar2(P(j), EWd)$;
$Q(j) \leftarrow GenQ(C(j), TNa, EWd)$;
$TDm = \sum_N (TDm(j) + TDm_\Delta(j))$;
$TEm = \sum_N (TEm(j) + TEm_\delta(j))$;
$TEm_{opt} \leftarrow GetOpt(TEm, TDm)$;
$b(j) = \max_{x_i \in Q(j)} EWd(x_i)$;
$m(j) = |Q(j)|$;
$return(N, Q(j), TEm_{opt}, TDm, b(j), m(j))$;

Fig. 5 Pseudo code of VAbM technique

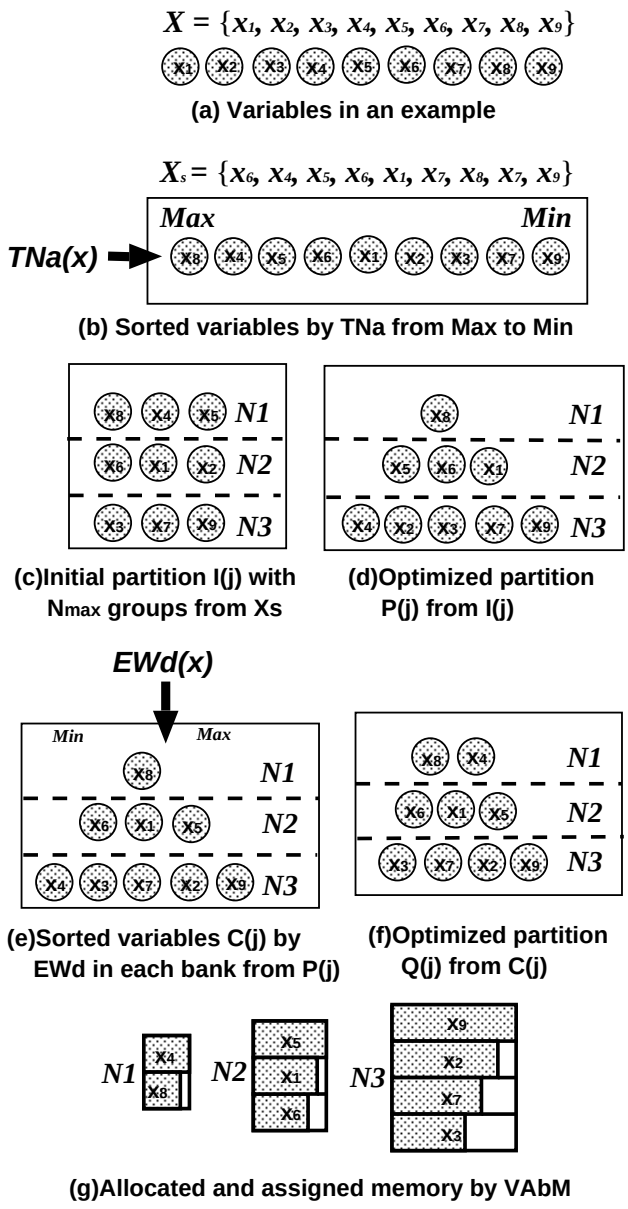

Fig. 6 VAbM Technique

and the memory delays, to find the optimal number of banks N and the assignment of those variables $Q(j)$ in each bank to optimize memory energy consumption. If we select the characteristics of the memory modules such as bit width $b(j)$ and word count $m(j)$ considering all possible patterns to optimize memory, it will be a NP-hard problem. An exhaustive-search algorithm to solve the memory assignment problem would have to first generate clusters of all combinations of compatible variables(variables that can share the same SRAM space) using lifetime analysis results and then generate all possible combinations of these clusters and pick the combination with total size fitting into the SRAM that minimize the total energy consumption. This procedure requires $O(2^{2^n})$time, which is unacceptably expensive, since the function $y = 2^{2^n}$ grow very rapidly, even for small values of n.

However, the memory assignment problem domain for practical applications are restricted to a smaller range of bit-widths(the bit-width of a memory module typically lies between 4 and 128) and the maximum number of memory banks constraint is smaller (usually $N_{Max} < 4$). Thus, we can apply an exhaustive search (enumeration) scheme to find the optimal solution. We propose VAbM technique, considering variable analysis results of lifetime, access frequency and effective data width, which is evaluated by our experiments shown in section 5.

For VAbM technique, we use the SRAM models shown in section 2, formulation (2), (3) and (5). We assume that we can use the arbitrary number of bit lines and word lines. Figure5 is the pseudo-code of VAbM technique composed of two steps. In the first step, variables are analyzed, effective data width LT, access frequency TNa and lifetime EWd of variables are reported. This information will be used to customize the memory architecture.

In the second step, memory allocation and assignment based on variable analysis results are performed, and energy consumption of the allocated and assigned memory is optimized. In our solution to the memory allocation and assignment problem, we first group vari-

ables that could share SRAM space into clusters using lifetime analysis results of variables to minimize memory storage. We formulate this problem as clique-partitioning problem [19] [20]. Then we customize memory for a given application considering access frequency and effective data width of variables. Variables with high frequency of access and small effective data width are good candidates to store in a small memory with few numbers of bit lines and word lines to reduce energy consumption of memory. However, in many applications the variables with small effective data width may have low access frequency. Because in real application, the access frequencies are far bigger than effective data width of variables ($TNa >> EWd$), we partition memory considering access frequency of variables first. For example, we have a given application program with the variables, $X = \{x_1, x_2, ..., x_{10}\}$ (assume that variables have been compacted by lifetime analysis results of variables) shown in Figure 6(a) and a given constraint number of memory banks which is 3 ($N_{Max} = 3$).

For VAbM technique, firstly we do not do any partition and we can compute the energy consumption $Emon$ for monolithic memory. Secondly, we add the access frequency $TNa(x_i)$ to each variable x_i in X, and then sort these variables, generate a sorted variable set X_s, which is arranged from maximum access count to minimum access count shown in Figure 6(b). Thirdly, we do the initial partition, namely the number of variables sorted in X_s is divided by the maximum number constraint of memory banks, N_{Max}, then we get $I(j)$ 6(c). Fourthly, optimize partition $I(j)$ by access count of variables. In the example, we move the variables by access count (from maximum to minimum) from $I(2)$ to $I(1)$ until the total energy consumption TEm is minimized or $P(1) = \emptyset$, then repeat the same thing with $I(3)$ and $I(2)$. At last, we get optimized memory banks configuration $P(j)$ 6(d). Fifthly, we sort the variables by effective data width $EWd(x_i)$ in $P(j)$ into $C(j)$ 6(e), and pick the maximum and minimum of $TEn(j)$, we mark them $C_{Emax}(j)$ and $C_{Emin}(j)$ respectively, then from $C_{Emax}(j)$ move the variables by effective data width $EWd(x_i$ (from minimum to maximum), until total energy consumption TEM is minimized, or $C_{Emax} = \emptyset$. Therefore, from $C(j)$, we get $Q(j)$ 6(f). And then, variables in $Q(j)$ are assigned to each bank, energy consumption $TEm(j)$ of different memory banks, overhead values of energy penalty $TEm_\delta(j)$ for adding banks and the total energy consumption TEm of memories are estimated, at last the optimized memory allocation and assignment is achieved shown in Figure6(g).

5. Experiments and Results

This section presents experiments and results to evaluate the proposed VAbM technique. We use real embedded applications as our benchmarks. Our target is to customize memory suitable to a given application. We assume that the physical model of the processor-based system with allocated and assigned memory is that the processor is placed facing the memory system. The memory address and data buses are between two levels of memory banks facing each other. Memory selection and decode logic is placed between memory and processor shown in figure 7. This is a simple physical model used as a basis for assuming the values for energy penalty coefficient $\delta = \{\delta(1), \delta(2), ..., \delta(N)\}$, and the maximum number of allocation banks N_{Max}.

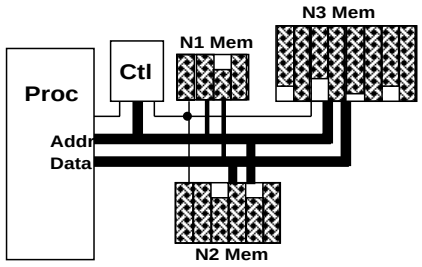

Fig. 7 Physical model of the allocated and assigned memory using VAbM

In our experiments, $N_{Max} = 3$ is assumed. This is a conservative bound on allocation and assignment. δ is assumed to $\delta = [0, 0.15, 0.10]$. Overhead penalty $\delta(2) = 0.15$ means that the overhead energy is 15% of the energy consumed by the monolithic memory, which is the largest because of a sizable penalty caused by the selection control logic, the routes of the buses and control wires from the unpartitioned memory to the partitioned solution. $\delta(3) = 0.10$ is still big, because when one bank is added from two banks to three banks, a bus stub to the right of the rest pair of banks is needed. The simplified physical model is obviously just one of the many possible choices. Our technique is completely independently from it. Furthermore, the values of δ we set is fairly conservative. In fact, the penalties can be tightened to use appropriate layout techniques,, and more aggressive partitioning ($Max > 3$) could be considered in a real design.

Table 1 shows the results of the experiments employed our low-energy memory design technique based on variable analysis(VAbM). To illustrate the effectiveness of our technique, we compare the experimental results to not only monolithic memory, but also memory designed by banking technique, which is usually used by most of memory designers. We use five real embedded applications as benchmarks, which are calculator, Lempel-Ziv algorithm, ADPCM encoder, MPEG-2 AAC audio decoder, and MPEG-2 video decoder. The first column $Emon$ shows the total energy consumption of monolithic memory. The next three columns show the results of memory banking technique. $Configuration$ shows the details on how the various memory banks are organized, in which bit width of

each memory bank is 32bits, TEb is the total energy consumption of memory banks. *Saving* shows the reduction compared to monolithic memory. The allocation and assignment results obtained by VAbM technique are listed in the last three columns, where Column $Configuration$ provides the details on how the various memory banks are organized, while TEm gives the total energy consumption of the optimized memory. Finally, column *Saving* reports the percentage of energy reduction for the optimal organization over the monolithic one. Our approach achieved drastically energy reduction up to 64.8%, the average energy savings is of 46.4%. The energy results also include the wiring and logic energy overhead given by δ. Compared to the memory designed by memory banking technique, our experimental results show that we can get up to 27.7% energy reduction, average 15.9% for the experimental applications shown in figure 8.

By considering not only access frequencies but also effective data width of variables, we combine the memory banking technique with variable analysis technique to perform memory allocation and assignment for a given application. The memory storage is managed more judiciously, resulting in significant energy reduction, without sacrificing cycles.

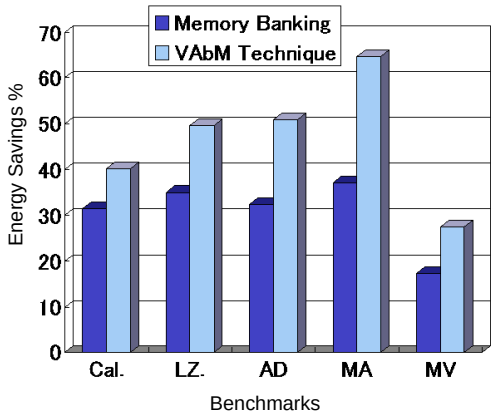

Fig. 8 Energy savings of benchmarks

6. Conclusions

In this paper, we proposed a low-energy memory organization technique based on variable analysis (VAbM), which presents the optimum solution under a given constraint of the maximum number of banks. The hardware and wiring overhead due to additional memory banks is properly taken into account as a penalty factor. We have demonstrated significant energy savings average about 46.4%, based on several real embedded applications with respect to monolithic memory and 27.7% to memory designed by memory banking technique. We will work on integration of customizing both processors and memories based on variable analysis for low-energy application-specific systems.

Acknowledgement

This research was partly supported by the Grant-in Aid for Scientific Research (B) (2) 12558029 and VCDS project of STARC.

References

[1] P. R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, E. Brockmeyer and A. Vandercappelle "Data and Memory Optimization Techniques for Embedded Systems", ACM Transactions on Design Automation of Electronic Systems, Vol 6, No. 2, pp. 149-206, April 2001.

[2] D. Lidsky and J. Rabaey "Low-power design of memory intensive functions", Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design, pp16-17, 1994.

[3] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachergaele, A. vandecapplle, "Custom Memory Management Methodology Exploration for Memory Optimization for Memory Optimization for Embedded Multimedia System Design", Kluwer, 1998.

[4] P. R. Panda, N. D. Dutt, F. Catthoor, A. Vandercappelle, E. Brockmeyer, C. Kulkarni, and D. Greef "Data Memory Organization and Optimizations in Application-Specific Systems", IEEE Design & Test of Computers, Vol 18, No. 3, pp. 56-68, MAY-JUNE 2001.

[5] T.Ishihara and H.Yasuura, "A Power Reduction Technique with Object Code Merging for Application Specific Embedded Processors", Proceedings of IEEE/ACM Design Automation and Test in Europe (DATE2000), pp.617–622, Mar. 2000.

[6] Sari L. Coumeri and Donald E. Thomas, "Memory Modeling for System Synthesis", Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design, 1998.

[7] Catherine H. Gebotys, "Low Energy Memory and Register Allocation Using Network Flow", IEEE Proc.of 34th. Design Automation Conference(DAC'97),1997.

[8] W. Shiue, C. Chakrabati, "Memory Exploration for Low Power, Embedded Systems", IEEE/ACM Proc.of 36th. Design Automation Conference(DAC'99),1999.

[9] A. Sudarsanam and S. Malik, " Memory bank and register allocation in software synthesis for ASIPs". Proc. of IEEE/ACM International Conference on Computer-Aided Design, pp388-392, Nov. 1995.

[10] M. A. Sagghir, P. Chow and C. G. Lee, " Exploiting dual data-memory banks in digital signal processors". Proc. of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, pp234-243, Oct. 1996.

[11] L. Benini, A. Macii, E. Maccii, M. Poncino, "Synthesis of Application-Specific Memories for Power Optimization in Embedded Systems" IEEE/ACM Proc.of 37th. Design Automation Conference, 2000.

[12] L. Benini, A. Macii, M. Poncino, "A Recursive Algorithm for Low-Power Memory Partitioning" Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design, 2000.

[13] T. Watanabe, R. Fujita, K.Yanagisawa, "Low-Power and High Speed Advantages of DRAM-Logic Integration for Multimedia Systems," IEICE Trans. on Electronics, Vol. E80-C, No.12, pp. 1523-1531, Dec. 1997.

[14] H. Yasuura, H. Tomiyama, A. Inoue and F. N. Eko, "Em-

Table 1 Optimized memory organization by VAbM technique

Application	Emon	Memory banking technique			Optimized memory by VAbM		
		Configuration	TEb	Saving	Configuration	TEm	Saving
Calculator	1.27 mJ	85rows 154rows 533rows	0.87 mJ	31.5%	85rows × 8b 154rows × 32b 533rows × 32b	0.76 mJ	40.2%
Lempel-Ziv	1.37 J	830rows 3rows 1663rows	0.89 J	35.0%	830rows × 13b 3rows × 15b 1663rows × 15b	0.69 J	49.6%
ADPCM	1.63 J	20rows 16rows 86rows	1.10 J	32.5%	20rows × 10b 16rows × 14b 86rows × 19b	0.80 J	50.9%
Mpeg2AAC	1.05 J	30rows 2374rows 4804rows	0.39 J	37.1%	30rows × 20b 2374rows × 32b 4804rows × 32b	0.37 J	64.8%
Mpeg2Video	145.1 kJ	26559rows 26557rows 28127rows	120.1 kJ	17.2%	26559rows × 8b 26557rows × 30b 28127rows × 32b	105.2 kJ	27.5%

bedded System Design Using Soft-core Processor and Valen-C", IPSJ.Info. Sci. Eng. vol. 14, pp.587-603, Sept. 1998.

[15] H. Yamashita, H. Yasuura, F. N. Eko, and Yun Cao, "Variable Size Analysis and Validation of Computation Quality", *Proc. of Workshop on High-Level Design Validation and Test*, HLDVT00, pp.95–100, Nov. 2000.

[16] Yun Cao, Hiroto. Yasuura, "A System-level Energy Minimization Approach Using Datapath Width Optimization", Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design, 2001.

[17] A.V.Aho, R. Sethi, and J. D. Ullman, "Compilers-Principles, Techniques and Tools", Addison-Wesley, 1986.

[18] A.Inoue, H.Tomiyama, T.Okuma, H.Kanbara, and H.Yasuura, "Language and Compiler for Optimizing Datapath Width of Embedded Systems", IEICE Trans. Fundamentals, Vol. E81-A, No.12, pp. 2595-2604, Dec. 1998.

[19] D. D. Gajski, N. D. Dutt, A. C-H Wu and S. Y-L Lin, "High-level synthesis introduction to chip and system design", Kluwer Academic Publishers Group, 1992.

[20] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph-Algorithm 457," Commun. ACM, 16(9): 575-577. September, 1973.

Yun Cao is a PhD candidate in Department of Computer Science and Communication Engineering, Kyushu University, Japan. She received her B.E. degree of Electronics and Information Engineering from Huazhong University of Science and Technology, China, in 1989. Her research interests are hardware/software codesign, low power/low energy system design and system design methodology. She is a student member of IPSJ, IEEE and ACM.

Hiroto Yasuura is a professor of Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, and the director of System LSI Center of Kyushu University, Fukuoka, Japan. He received the B.E., M.E., and PhD. degrees in computer science from Kyoto University. His current interests include parallel computer architectures, hardware algorithms for VLSI, VLSI CAD, and system design methodology. He is a member of IPSJ, ACM, and IEEE.