Data Memory Design Considering Effective Bitwidth for Low-Energy Embedded Systems

Yun Cao Hiroyuki Tomiyama* Takanori Okuma Hiroto Yasuura
Department of Computer Science and Communication Engeering, Kyushu University
Kasuga Koen 6-1, Fukuoka 816-8580, Japan
{cao,okuma,yasuura}@c.csce.kyushu-u.ac.jp
*Institute of Systems & Information Technologies/KYUSHU
Fukuoka SRP Center Building 7F, 2-1-22, Momochihama, Fukuoka 814-0001, Japan
{tomiyama}@isit.or.jp

ABSTRACT

This paper presents a novel low-energy memory design technique, considering effective bitwidth of variables for application-specific systems, called VAbM technique. It targets the exploitation of both data locality and effective bitwidth of variables to reduce energy consumed by redundant bits. Under constraints of the number of memory banks, the VAbM technique use variable analysis results to perform allocating and assigning on-chip RAM into multiple memory 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 18.4% over memory designed by banking technique.

Categories and Subject Descriptors

B.3.2 [**Memory Structures**]: Design Styles; J.2 [**Computer-aided Engineering**]: Computer-aided Design

General Terms

Design, Algorithms

1. INTRODUCTION

Memory-processor integration on System-on-a-Chip 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. 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, especially, for data-dominated applications. Embedded processor-based systems allow customization of memory configuration based on application-specific requirements [1].

Some researches have analyzed the power dissipation of different memory architectures for a given application. [2] presented power reduction technique for instruction memories in application specific systems. [3] presented a methodology for developing models of on-chip SRAM memory organization. [4] 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.

On-chip caches are well known architectural optimization technique for memory design [5]. Our work moves from the observation that cache memories are not the most power-efficient architecture. Data 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 is a valid alternative to caches. In this architecture, the most frequently accessed addresses are statically mapped onto SRAM to guarantee power and performance efficiency. The main difference between the SRAM and data cache is that the SRAM guarantees a single-cycle access-time, whereas access to the cache is subject to cache misses.

Our approach is close to [6], which focused on 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.

Our work differs from the ones above in that we focuse on memory allocation and assignment for low-energy on-chip RAM with respect to the memory utilization considering not only memory access frequency but also effective bitwidth of variables. We present an exploration technique for determining efficient on-chip memory configuration, characterized by the size of 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. This paper focuses only on the data memory optimization 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 describes variable analysis. Section 3 presents our technique for low-energy memory design based on variable analysis,

ISSS'02, October 2–4, 2002, Kyoto, Japan.

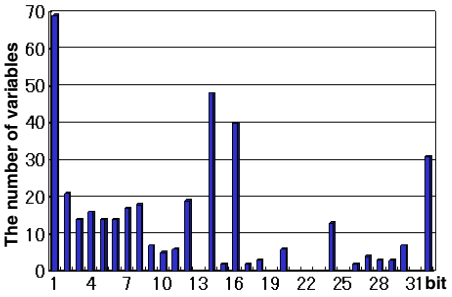

Figure 1: Variable Distribution for Effective Bitwidth in MPEG-2 Decoder

calledd VAbM technique. Experiments and results are reported in Section 4. Finally, Section 5 concludes and gives our future work.

2. VARIABLE ANALYSIS

In application-specific system design, to some extent, behavior of the application is statically known, although it has some dependence on input data. Therefore, analyzing the application program and using the characteristics to further optimization for application-specific systems is necessary. By analyzing effective bitwidth of variables we can reduce unused bits to reduce memory storage requirement and more over by analyzing the lifetime of variables, we can make efficient use of memory storage by variables with disjoint lifetime sharing same memory space. In addition, by analyzing access frequency of variables, we can use data locality efficiently to reduce memory access energy.

2.1 Data Width Analysis

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

In this paper, we define *effective bitwidth* as the smallest size which can hold both maximum and minimum values of a variable. We use two methods to analyze effective bitwidth of variables [7] [8]. One is static analysis; the other is simulation-based dynamic analysis.

We analyzed the C source program of MPEG-2 video decoder using our developed variable size analyzer and got the variable analysis results of effective bitwidth depicted in Figure 1. This figure shows that there are a lot of variables having many unused bits in MPEG-2 decoder, which originally declared as "int" type.

2.2 Lifetime Analysis

The lifetime of a variable is defined as the period between its definition and last use [9]. 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 [10], so that we can cluster them to share same memory space. Therefore, by analyzing lifetime of variables, we can make efficient use of memory.

2.3 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 an 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 got the Figure 2, which shows the profiled results for the access frequencies of variables with effective bitwidth in MPEG-2 decoder using our profiler. From the figure, we can see that there are "hot spots" for variable access in MPEG-2 decoder.

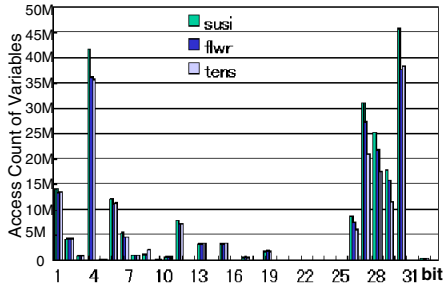

Figure 2: Variable Access Count of MPEG-2 Decoder

3. LOW-ENERGY MEMORY DESIGN

Memory design can be divided into two steps. First, several memories are selected 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.

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 bitwidth 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 target is to determine the memory allocation and assignment by mapping each variable into on-chip SRAM so as to minimize total energy consumption of memory.

The target system is assumed as follows: Data memories are organized by several on-chip SRAMs having different size with different number of bit lines and different word lines, a hierarchical memory.

The memory model is shown in Figure 3. 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 energy savings.

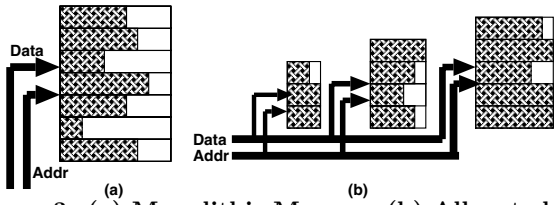

Figure 3: (a) Monolithic Memory (b) Allocated and Assigned Memory Using Our Technique

As a concrete example of a memory architecture that can be modeled with the template of Figure 3, 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 cycle for read accessing, for N1 memory is approximately 337pJ; for N2 memory access is 852pJ; and for N3 access is 1132pJ respectively.

The proposed optimization technique is based on the following observations:

* 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, in which many bits are never used during program executions.
* Variables with disjoint lifetimes can share same memory space.

3.1 Basic Approach

The main purpose of memory allocation and assignment is to minimize overall energy cost within performance and memory size constraints. Hierarchical organizations reduce memory energy consumption by exploiting not only the nonuniformities in access frequencies but also effective bitwidth of variables. We generate a hierarchical 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 consists of the following phases:

* Phase 1: For a given application program, analyze variables, report effective bitwidth, access frequency and lifetime of variables.
* Phase 2: Using analysis results including effective bitwidth, access frequency and lifetime of variables, perform memory allocation and assignment considering not only the number of words but also the bit width of memory, in order to reduce energy consumption of memories.

The details will be explained in section 3.4.

3.2 Energy Cost Metrics

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. Although our technique can also be used for DRAMs, this work focuses on on-chip SRAMs.

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 Formula(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 Formula(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 Formula(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:

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

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

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.

The models demonstrate 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.

The energy cost function employed for estimating memory energy is shown as follows. The total energy consumption of memories is the summation of the memory banks.

$$TEm = \sum_{N} (TEm(j) + TEm_{\delta}(j)) \quad (4)$$

$$TEm(j) = \sum_{|Q(j)|} E(i,j) \quad (5)$$

$$E(i,j) = e_r(j) \cdot TNa_r(x_i) + e_w(j) \cdot TNa_w(x_i) \quad (6)$$

$$TEm_{\delta}(j) = Emon \cdot \delta(j) \quad (7)$$

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 a monolithic memory
$E(i,j)$: Energy consumption of x_i for read and write access to memory bank j
X : A finite set of variables in a given application program, $X = \{x_1, x_2, ..., x_i, ..., x_n\}$
x_i : A variable, $x_i \in X$
n: The number of variables in a given program
$e_r(j)(e_w(j))$: Energy consumption per read (write) access for bank j
$TNa_r(x_i)(TNa_w(x_i))$: The number of read (write) accesses for variable x_i
$Q(j)$: The set of variables assigned into bank j $Q(j) \subseteq X$
$\delta(j)$: Overhead coefficient for added bank j, caused by addressing complexity.

3.3 Problem Formulation

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

Assumption 1

* The maximum number N of patitioned memory banks is given, and we assume that there is not a cache in the system.

Assumption 2

* If the effective bitwidth of the variable x_i is bigger than the bit width of the memory bank j, variable x_i will not be assigned into memory bank j. Namely, a variable will not be divided into two words.

Assumption 3

* Several variables will not be merged as one word. For example, if the effective bitwidth of variable x_1 and x_2 are 16bits respectively, and they will be assigned into memory bank 1, which has bit width of 32bits. However, x_1 and x_2 are assigned as one word (32bits) separately.

Notation

* $EWd = \{EWd(x_1), EWd(x_2), ..., EWd(x_i), ..., EWd(x_n)\}$: A set of effective bitwidth for variables X
* $TNa = \{TNa(x_1), TNa(x_2), ..., TNa(x_i), ..., TNa(x_n)\}$: A set of total memory access for variables X
* $LT = \{LT(x_1), LT(x_2), ..., LT(x_i), ..., LT(x_n)\}$: A Set of Lifetime for variables X
* X, x_i, $\delta(j)$, TEm, N, $TEm(j)$, $TEm_\delta(j)$, $Q(j)$: Defined in previous subsection.

Problem Definition

Given X, EWd, TNa, LT and N
To find N and $Q(j)$
So that $TEm = \sum_N (TEm(j) + TEm_\delta(j))$ is minimized
Subject to

$$Q(j) \subseteq X$$
$$X = Q(1) \bigcup Q(2) ... \bigcup Q(N)$$
$$Q(1) \bigcap Q(2) ... \bigcap Q(N) = \phi$$

Allocated and assigned memory

With N memory 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

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

ILP Formulation

We present an ILP(Integer Linear Programming) formulation for the memory allocation and assignment problem. The ILP model is used to find the number of modules and the size(bit width and the number of words)of each module such that the energy consumption is minimum under the constraints of the maximum number of partitioned memory banks. We refer to the set of variables that are assigned into each modules as a variable grouping. The first step in the procedure is to identify the permissible variable groupings. The variables in a variable grouping are assigned to the same module.

The next step is to derive the ILP model. We give notations used in the formulation.

* $a(i,j)$: Integer variables, $a(i,j) = 1$, If variable x_i is assigned into memory bank j; otherwise, $a(i,j) = 0$
* U: A constant of positive integer, $U > n$
* $b(j)$: Integer variables, $b(j) = 1$, If memory bank j is selected; otherwise, $b(j) = 0$. Namely, $b(j) = 1$, when $\sum_i a(i,j) \geq 1$, $b(j) = 0$, when $\sum_i a(i,j) = 0$
* $\sum_i a(i,j) - U \cdot b(j) \leq 0$
* $\sum_i a(i,j) - b(j) \geq 0$

Input:
source program : AP
(variables : $x_i \in X = \{x_1, \ldots, x_n\}$, $1 \leq i \leq n$)
input data : D_{in}
the maximum number of banks : N
overhead coefficient : $\delta(j)$ ($1 \leq j \leq N$)
Variable:
the subset of variables $Q(j) \subseteq X$
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
$analyzer(AP, D_{in})$;
$LT = \{LT(x_1), \ldots, LT(x_n)\}$,
$TNa = \{TNa(x_1), \ldots, TNa(x_n)\}$,
$return(EWd = \{EWd(x_1), \ldots, EWd(x_n)\})$;
Step 2 : Memory Allocation and Assignment
$X_c \leftarrow CompactVar(X, LT)$;
for each $x_i \in X_c$
$a(i,j) = 0$;
end for
$TE_{opt} = Emon$;
$Q(j) = \phi$;
for (N)
while (leads to reduction of TE_{opt})
for $(i = 0...n-1)$
$Cost = OBJ(i,j)$
if $(Cost < TE_{opt})$
$TE_{opt} = Cost$;
$Q(j) \leftarrow x_i$;
end for
$b(j) = max_{x_i \in Q(j)} EWd(x_i)$;
$m(j) = |Q(j)|$;
end for
$return(N, Q(j), TEm_{opt}, b(j), m(j))$;

Figure 4: Pseudo Code of the Algorithm

Objective function:
Minimize: $Cost = OBJ(i,j) = \sum_i \sum_j E(i,j) \cdot a(i,j)$
Constraints:
(1)$1 \leq i \leq n, 1 \leq j \leq N$;
(2)the number of memory banks: $\sum_j b(j) \leq N$;
(3)memory size: $\sum_i a(i,j) \leq |Q(j)|$, for all j;
For a given N, find a set of a(i,j) which minimize objective function.

3.4 VAbM Technique

This section presents a technique for application-specific memory organization in detail. VAbM receives a 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, 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 an NP-hard problem. A simple exhaustive search 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 RAM that minimizes the total energy consumption. This procedure requires $O(2^{2^n})$time complexity, which is unacceptably expensive, since the function $y = 2^{2^n}$ grows 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 $N \leq 4$). We use greedy algorithm to find the optimal solution. We propose VAbM technique, considering variable analysis results of lifetime, access frequency and effective bitwidth, which is evaluated by our experiments shown in Section 4.

For VAbM technique, we use the SRAM models shown in Section 3.2, Formula (2) and (3). We assume that we have a memory library, which have modules with the arbitrary number of bit lines and word lines. Figure 4 is the pseudocode of VAbM technique composed of two steps. In the first step, variables are analyzed, effective bitwidth EWd, access frequency TNa and lifetime LT 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 variables that can share SRAM space into clusters using lifetime analysis results of variables to minimize memory storage. We formulate this problem as clique-partitioning problem [11]. Then we customize memory for a given application considering access frequency and effective bitwidth of variables. Variables with high frequency of access and small effective bitwidth 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 bitwidth may have low access frequency. Because in real application, the access frequencies are far bigger than effective bitwidth of variables ($TNa >> EWd$), we partition memory considering access frequency of variables first.

We use the greedy algorithm shown in Figure 4. The complexity of this heuristic algorithm is $O(n)$. The input to the algorithm for memory optimization is a set of variables X, where each variable $x_i \in X$ is characterized by its access frequency, its effctive bitwidth and its location $a(i, j)$. All $a(i, j)$ are set to zero at first step. This means that all the variables are assigned into one memory (a monolithic memory). Next, the algorithm provisionally relocate a variable x_i from the monolithic memory to a partitioned memory bank j (a memory module selected from library). After calculating $Cost$, the provisionally located variable x_i is moved back to former location. This process is operated for each variable. After that, the algorithm selects a variable which minimizes the $Cost$, and the selected variable is moved to the memory bank j. Variables located in the monolithic memory is successively moved in this way while the $Cost$ is reduced. If the cost becomes not to be improved, the algorithm stops, at last the optimized memory allocation and assignment is achieved.

4. 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 5. 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.

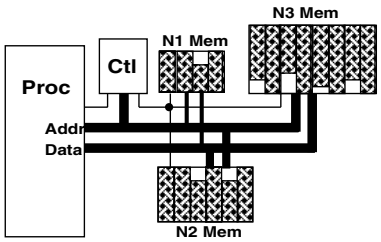

Figure 5: Physical model of the allocated and assigned memory using VAbM

In our experiments, $N = 3$ 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 ($N > 3$) 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 shown in the first column. The second 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 the 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, the column *Saving* reports the percentage of energy reduction for the optimal organization over the monolithic one. Our approach achieved energy reduction up to 64.8%, the average energy savings

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	62.8%	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%

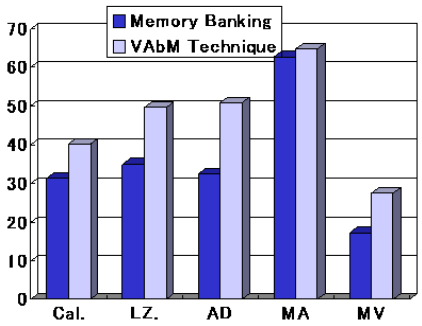

Figure 6: Energy savings of benchmarks

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 18.4% energy reduction, average 10.8% for the experimental applications shown in Figure 6.

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

5. CONCLUSIONS

This paper proposed a low-energy memory design technique based on variable analysis(VAbM) considering effective bitwidth of variables, 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. Experimental results on several real embedded applications demonstrated significant energy savings about 46.4% on average, with respect to a monolithic memory and 10.8% to memory used memory banking technique. We will work on integration of customizing both processors and memories based on variable analysis for low-energy application-specific systems.

6. ACKNOWLEDGMENTS

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

7. 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] T.Ishihara and H.Yasuura, "A Power Reduction Technique with Object Code Merging for Application Specific Embedded Processors". Proc. of Design Automation and Test in Europe, pp.617–622, Mar. 2000.

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

[4] Catherine H. Gebotys, "Low Energy Memory and Register Allocation Using Network Flow". Proc.of 34th. Design Automation Conference, June 1997.

[5] W. Shiue, C. Chakrabati, "Memory Exploration for Low Power, Embedded Systems". Proc.of 36th. Design Automation Conference, June 1999.

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

[7] 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, pp.95–100, Nov. 2000.

[8] Yun Cao, Hiroto. Yasuura "A System-level Energy Minimization Using Datapath Optimization". Proc. of International Symposium on Low Power Electronics and Design, August 2001.

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

[10] 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.

[11] 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.