Low-Power Processors Require Effective Memory Partitioning

Leonardo Steinfeld, Marcus Ritt, Fernando Silveira, Luigi Carro

Low-power processors require effective memory partitioning

Leonardo Steinfeld[1], Marcus Ritt[2], Fernando Silveira[1], and Luigi Carro[2]

[1] Instituto de Ingenieria Electrica, Facultad de Ingenieria, Universidad de la Republica, Uruguay.
[2] Instituto de Informatica, Universidade Federal do Rio Grande do Sul, Brasil.

Abstract. The ever increasing complexity of embedded systems demands for rising memory size, and larger memories increase the power drain. In this work, we exploit banked memories with independent low-leakage retention mode in event-driven applications. The resulting energy saving for a given number of banks is close to the maximum achievable value, since the memory banks access pattern of event-driven applications presents a high temporal locality, leading to a low saving loss due to wake-up transitions. Results show an energy reduction up to 77.4% for a memory of ten banks with a partition overhead of 1%.

Keywords: banked memory, event-driven applications, power management, wireless sensor network

1 Introduction

In the last years, there has been a lot of research dealing with processing power optimization resulting in a variety of ultra-low power processors. These processors pose a primary energy limitation for SRAM, where the embedded SRAM consumes most of the total processor power [1]. Partitioning a SRAM memory into multiple banks that can be independently accessed reduces the dynamic power consumption, and since only one bank is active per access, the remaining idle banks can be put into a low-leakage sleep state to also reduce the static power [2]. However, the power and area overhead due to the extra wiring and duplication of address and control logic prohibits an arbitrary fine partitioning into a large number of small banks. Therefore, the final number of banks should be carefully chosen at design time, taking into account this partitioning overhead. The memory organization may be limited to equally-sized banks, or it can allow any bank size. Moreover, the strategy for the bank states management may range from a greedy policy (as soon as a bank memory is not being accessed it is put into low leakage state) to the use of more sophisticated prediction algorithms [3].

Memory banking has been applied for code and data using scratch-pad and cache memories in applications with high performance requirements (e.g. [2],[4]). We follow the methodology employed in [4], in which a memory access trace is used to solve an optimization problem for allocating the application memory

divided in blocks to memory banks. However, to the best of our knowledge, this is the first time SRAM banked memories are considered for event-driven applications code, and the use of such characteristics leads to meaningful power savings, as it will be shown.

The main contribution of this work is to show that, thanks to our new problem formulation, one can find the optimum partitioning of memory banks in the very common event-driven applications. We derive expressions for energy savings in the case of equally sized banks based on a detailed model for different power management strategies. The maximum achievable energy saving is found, and the limiting factors are clearly determined. We show that it is possible to find a near optimum number of banks at design time, irrespective of the application and the access pattern to memory, provided that the energy memory parameters are given, such as energy consumption characteristics and the partition overhead as a function of the number of banks. We show that using our approach in a banked memory leads to aggressive (close to 80%) energy reduction in event-driven applications.

The remainder of this paper is organized as follows. In Section 2, we present a memory energy model, and in Section 3 we derive expressions for the energy savings of a banked memory. The experiments are presented in Section 4 and in Section 5 we discuss the results. Finally Section 6 contains concluding remarks.

2 Banked memory energy model

In this section we present a memory energy model for deriving expressions for the energy consumption of an equally-sized banked memory with different power management strategies.

2.1 Memory energy model

The static power consumed by a SRAM memory depends on its actual state: ready or sleep. During the ready state read or write cycles can be performed, but not in the sleep state. Since the memory remains in one of these states for a certain amount of cycles, the static energy consumed can be expressed in terms of energy per cycle (E_{rdy} and E_{slp}) and number of cycles in each state. Each memory access, performed during the ready state, consumes a certain amount of energy (E_{acc}). The ready period during which memory is accessed is usually called the active period, and the total energy spent corresponds to the sum of the access and the ready energy ($E_{act} = E_{acc} + E_{rdy}$), i.e. the dynamic and static energy. On the other hand, the ready cycles without access are called idle cycles, consuming only static energy ($E_{idl} = E_{rdy}$). Each state transition from sleep to active (i.e. the wake-up transition) has an associated energy cost (E_{wkp}) and a latency, considered later. Based on the parameters defined above, the total energy consumption of a memory can be defined as

$$E = E_{act}n_{act} + E_{idl}n_{idl} + E_{slp}n_{slp} + E_{wkp}n_{wkp}, \tag{1}$$

Table 1. Memory energy consumption coefficients.

E_{act}	E_{idl}	E_{slp}	E_{wkp}
1.78×10^{-6}	3.28×10^{-7}	3.28×10^{-8}	7.95×10^{-6}

where n_{act}, n_{idl} and n_{slp} are the sum of the cycles in which the memory is in active, idle and in sleep state respectively, and n_{wkp} is the number of times the memory switches from sleep to active state.

The energy values in Eq. (1) depend on the size of the memory, and generally energy is considered proportional to it [2]. Using the CACTI tool [5], we simulated a pure RAM memory, one read/write port, 65 nm technology and a high performance ITRS transistor type, varying its size from 512 B to 256 KB. CACTI outputs the dynamic and leakage energy, corresponding to the access and idle of our model. The active energy is directly computed (dynamic plus leakage). The access, active, and idle energy were fitted to a linear function as a function of the memory size to determine the energy coefficients. The energy consumed per cycle in the sleep state is a fraction of the idle energy, since we suppose that a technique based on reducing the supply voltage is used to exponentially reduce the leakage [6]. We considered a reduction factor of leakage in sleep state of 0.1, which is generally accepted in the literature [7]. Finally, before a memory bank could be successfully accessed, the memory cells need to go back from the data retention voltage to the ready voltage, which involves the loading of internal capacitances. Since the involved currents in this process are similar to those in an access cycle, the associated wake-up energy cost is proportional to the access energy, ranging the proportionality constant from about 1 [8] to hundreds [9]. We adopt an intermediate value of 10. Table 1 shows the different coefficients used in the remainder of this work.

Finally, partitioning a memory in N equally sized banks reduce the energy by N,

$$E_k = \frac{\mathbf{E}_k}{N}. \tag{2}$$

for $k \in \{act, idl, slp, wkp\}$, where $\mathbf{E}_k$ is the corresponding energy consumption per cycle of the whole memory.

3 Energy savings

In this section we derive expressions for the energy savings of a memory of equally sized banks for two different management schemes. The first general expression corresponds to any power management by means of which a bank may remain in idle state even if it is not accessed. The decision algorithm may range from a simple fixed time-out policy to dynamic and sophisticated prediction algorithms. The second expression correspond to the simplest policy, greedy, in which a bank is put into sleep state as soon as it is not being accessed, and is determined as a special case of the former.

The total energy consumption per cycle of the whole banked memory after n cycles have elapsed is

$$\bar{E}_N = E_{act} \sum_{i=1}^{N} \frac{n_{act_i}}{n} + E_{idl} \sum_{i=1}^{N} \frac{n_{idl_i}}{n} + E_{slp} \sum_{i=1}^{N} \frac{n_{slp_i}}{n} + E_{wkp} \sum_{i=1}^{N} \frac{n_{wkp_i}}{n}, \quad (3)$$

Since the total number of cycles is $n = n_{act_i} + n_{idl_i} + n_{slp_i}$ for all banks, and that there is only one bank active per cycle

$$\sum_{i=1}^{N} n_{act_i} = n \quad (4)$$

then we obtain

$$\bar{E}_N = E_{act} + (N-1)\, E_{slp} + (E_{idl} - E_{slp}) \sum_{i=1}^{N} \frac{n_{idl_i}}{n} + E_{wkp} \sum_{i=1}^{N} \frac{n_{wkp_i}}{n}. \quad (5)$$

The first two terms of the sum are the consumption of having only one bank in active state and the remaining $N-1$ banks in sleep state. The third term, related to the idle energy, depends on the fraction of idle cycles performed by each bank i. The last term of the sum represents the wake-up energy as a function of the average wake-up rate of each memory bank, that is the average number of cycles elapsed between two consecutive bank transitions from sleep to active (for example, one transition in 1000 cycles).

We define the energy savings of a banked memory as the relative deviation of the energy consumption of a equivalent single bank memory ($E_1 = NE_{act}$, always active)

$$\delta E = \frac{NE_{act} - \bar{E}_N}{NE_{act}}. \quad (6)$$

The energy saving of a banked memory of N uniform banks is

$$\delta E_N = \frac{N-1}{N} \left(1 - \frac{E_{slp}}{E_{act}}\right) - \frac{1}{N} \left(\frac{E_{idl} - E_{slp}}{E_{act}}\right) \sum_{i=1}^{N} \frac{n_{idl_i}}{n} - \\ - \frac{1}{N} \frac{E_{wkp}}{E_{act}} \sum_{i=1}^{N} \frac{n_{wkp_i}}{n}. \quad (7)$$

If a greedy power management is considered a memory bank it is put into sleep state as soon as is not being accessed, hence there is no idle cycles and Eq. (7) simplifies to

$$\delta E_N^{grdy} = \frac{N-1}{N} \left(1 - \frac{E_{slp}}{E_{act}}\right) - \frac{1}{N} \frac{E_{wkp}}{E_{act}} \sum_{i=1}^{N} \frac{n_{wkp_i}}{n}. \quad (8)$$

In this case, the application blocks allocation to memory banks must minimize the accumulated wake-up rate in order to maximize the energy saving. Note that the energy saving does not depend on the access profile among the banks, since the access to every bank costs the same as all banks have the same size. Still, the allocation of blocks to banks must consider the constraints of the banks size. Finally, the energy saving can be improved by increasing N and at the same time keeping the accumulated wake-up rate low. The maximum achievable saving corresponds to the sleep to active rate, which is equivalent to have the whole memory in sleep state. Even so, the partition overhead limits the maximum number of banks.

Compared to Eq. (8), the general expression Eq. (7) has an additional term, which is related to the energy increase caused by the idle cycles. This does not mean that the energy saving is reduced, since the accumulated wake-up ratio may decrease.

3.1 Effective energy saving

As mentioned previously, the wake-up transition from sleep to active state of a bank memory has an associated latency. This latency forces the microprocessor to stall until the bank is ready. The microprocessor may remain idle for a few cycles each time a new bank is waken up, incrementing the energy drain. This extra microprocessor energy can be included with the bank wake-up energy and for simplicity we will not consider it explicitly. Moreover, if the wake-up rate is small and the active power of the microprocessor is much higher than idle power, this overhead can be neglected. Additionally, the extra time due to the wake-up transition is not an issue in low duty-cycle applications, since simply slightly increases the duty-cycle.

On the other hand, the partitioning overhead must be considered to determine the effective energy saving. A previous work had characterized the partitioning overhead as a function of the number of banks for a partitioned memory of arbitrary sizes [9]. In that case the hardware overhead is due to an additional decoder (to translate addresses and control signals into the multiple control and address signals), and the wiring to connect the decoder to the banks. As the number of memory banks increases, the complexity of the decoder is roughly constant, but the wiring overhead increases [9]. The partition overhead is proportional to the active energy of an equivalent monolithic memory and roughly linear with the number of banks, as can be clearly seen by inspecting the data of the aforementioned work (3.5%, 5.6%, 7.3% and 9% for a 2-, 3-, 4-, and 5-bank partitions, resulting in an overhead factor of approximately 1.8% per bank). Consequently, the relative overhead energy can be modeled as:

$$\delta E_N^{ovhd} = k_{ovhd} N. \tag{9}$$

In this work, the memory is partitioned into equally-sized banks. As result the overhead is expected to decrease leading to a lower value for the overhead factor.

3.2 Energy savings limits

The energy savings in the limit, as the wake-up and idle contributions tend to zero, is

$$\delta E_N^{max} = \frac{N-1}{N}\left(1 - \frac{E_{slp}}{E_{act}}\right). \tag{10}$$

If the partition overhead is considered (Eq. 9), the maximum effective energy saving is

$$\delta E_{N,eff}^{max} = \frac{N-1}{N}\left(1 - \frac{E_{slp}}{E_{act}}\right) - k_{ovhd}N. \tag{11}$$

$\delta E_{N,eff}^{max}$ is maximized for

$$N_{opt} = \sqrt{\frac{1}{k_{ovhd}}\left(1 - \frac{E_{slp}}{E_{act}}\right)}. \tag{12}$$

4 Experiments

In this section we present experiments comparing the predicted energy savings by our model to the energy savings obtained by solving an integer linear program (ILP).

The criteria for selecting the case study application were: public availability of source files, realistic and ready-to-use application. We chose a wireless sensor network application (data-collection) from the standard distribution of TinyOS (version 2.1.0) [3]. The application, MultihopOscilloscope, was compiled for nodes based on a MSP430 microcontroller[4]. Each node of the network periodically samples a sensor and the readings are transmitted to a sink node using a network collection protocol.

We simulated a network composed of 25 nodes using COOJA[10] to obtain a memory access trace of one million cycles or time steps.

For the sake of simplicity, the block set was selected as those defined by the program functions and the compiler generated global symbols (user and library functions, plus those created by the compiler). The size of the blocks ranges from tens to hundreds of bytes, in accordance with the general guideline of writing short functions, considering the run-to-completion characteristic of TinyOS and any non-preemptive event-driven software architecture. The segments size of the application are 3205 bytes of text, 122 and 3534 bytes of zero-valued and initialized data respectively. The number of global symbols is 261.

The problem of allocating the code to equally sized banks was solved using an ILP solver for up eight banks, for the greedy power management using a segment

[3] www.tinyos.net
[4] www.ti.com/msp430

Table 2. Optimum number of banks as a function of partition overhead.

$k_{ovhd}(\%)$	0	1	2	3	5
N_{opt}	∞	10	7	6	4
$\delta E^{max}_{N,eff}(\%)$	97.1	77.4	69.2	62.9	52.8

trace of 5000 cycles. The total memory size was considered 10% larger than the application size, to ensure the feasibility of the solution. The average energy consumption is calculated considering the whole trace using the memory energy model and the block-to-bank allocation map. The energy saving is determined comparing with a memory with a single bank with no power management.

5 Results and Discussion

The optimum number of banks estimated using Eq. (12) (after rounding) as a function of k_{ovhd} (1%, 2%, 3% and 5%) is shown in Table 2. The energy savings is limited by the partition overhead, reaching a value of 77.4% for an overhead of 1%. The energy saving limit, as the partition overhead tends to zero and N to infinity, is 97.1% (the corresponding value of $1 - E_{slp}/E_{act}$).

Table 3 compares the energy saving results as a function of the number of banks and the partition overhead. It can be observed that the maximum energy saving for greedy strategy with 2%, 3% and 5% of partition overhead is achieved for seven, six and five banks respectively (both marked with a gray background). The optimum number of banks for an overhead of 5% differs from what arises in the previous limit case (see Table 2). This means that the saving loss due to wake-up transitions shifts the optimum number of banks. For a given partition overhead, the energy saving for the estimated optimum number of banks is within about 4% of the saving limit (comparing saving values in Table 2 to the corresponding values in Table 3, number of banks: four, six and seven, and partition overhead: 5%, 3% and 2% respectively). This difference came from the wake-up transitions losses. In this case study, due to its even-driven nature, the code memory access patterns are caused by external events. Each event triggers a chain of function calls starting with the interrupt subroutine. This

Table 3. Energy saving for greedy power management.

greedy		number of banks						
		2	3	4	5	6	7	8
k_{ovhd} (%)	1	43.7	58.1	64.8	68.9	71.4	72.9	73.7
	2	41.7	55.1	60.8	63.9	65.4	65.9	65.7
	3	39.7	52.1	56.8	58.9	59.4	58.9	57.7
	5	35.7	46.1	48.8	48.9	47.4	44.9	41.7

chain may include the execution of subsequent functions calls starting with a queued handler function called by a basic scheduler. The allocation of highly correlated functions to the same bank leads to a bank access pattern with a high temporal locality. Hence, the total wake-up fraction across the banks is very low. This explain the modest difference with the limit value.

6 Conclusions

We have found that aggressive energy savings can be obtained using a banked memory, up to 77.4% (estimated) for a partition overhead of 1% with a memory of ten banks, and 65.9% (simulated) for a partition overhead of 2% with a memory of seven banks. The energy saving is maximized by properly allocating the program memory to the banks in order to minimize the accumulated wake-up rate. The saving increases as a function of the number of banks, and is limited by the partition overhead. The derived model gives valuable insight into the particular factors (coming from the application and the technology) critical for reaching the maximum achievable energy saving. Moreover, at design time the optimum number of banks can be estimated, considering just energy memory parameters. The resulting energy saving for a given number of banks is close to the derived limit, since event-driven applications present access patterns to banks with a high temporal locality.

References

1. Verma, N.: Analysis Towards Minimization of Total SRAM Energy Over Active and Idle Operating Modes. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **19**(9) (2011) 1695–1703
2. Golubeva, O., Loghi, M., Poncino, M., Macii, E.: Architectural leakage-aware management of partitioned scratchpad memories. In: DATE '07: Proceedings of the conference on Design, automation and test in Europe, San Jose, CA, USA, EDA Consortium (2007) 1665–1670
3. Calimera, A., Macii, A., Macii, E., Poncino, M.: Design Techniques and Architectures for Low-Leakage SRAMs. Circuits and Systems I: Regular Papers, IEEE Transactions on **59**(9) (2012) 1992–2007
4. Ozturk, O., Kandemir, M.: ILP-Based energy minimization techniques for banked memories. ACM Trans. Des. Autom. Electron. Syst. **13**(3) (July 2008) 1–40
5. Thoziyoor, S., Ahn, J.H., Monchiero, M., Brockman, J.B., Jouppi, N.P.: A Comprehensive Memory Modeling Tool and Its Application to the Design and Analysis of Future Memory Hierarchies. In: 2008 International Symposium on Computer Architecture, Washington, DC, USA, IEEE (June 2008) 51–62
6. Qin, H., Cao, Y., Markovic, D., Vladimirescu, A., Rabaey, J.: SRAM leakage suppression by minimizing standby supply voltage. In: SCS 2003. International Symposium on Signals, Circuits and Systems. Proceedings (Cat. No.03EX720), Los Alamitos, CA, USA, IEEE Comput. Soc. (2004) 55–60
7. Rabaey, J.: Low power design essentials. Springer Verlag (2009)

8. Calimera, A., Benini, L., Macii, A., Macii, E., Poncino, M.: Design of a Flexible Reactivation Cell for Safe Power-Mode Transition in Power-Gated Circuits. IEEE Transactions on Circuits and Systems I: Regular Papers **56**(9) (September 2009) 1979–1993
9. Loghi, M., Golubeva, O., Macii, E., Poncino, M.: Architectural Leakage Power Minimization of Scratchpad Memories by Application-Driven Sub-Banking. IEEE Transactions on Computers (2010)
10. Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., Marrón, P.J.: COOJA/MSPSim: interoperability testing for wireless sensor networks. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. Simutools '09, ICST, Brussels, Belgium, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2009) 1–7