Scratchpad Memory : A Design Alternative for Cache On-chip memory in Embedded Systems

Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan, Peter Marwedel
banakar | mbala@cse.iitd.ernet.in
Indian Institute of Technology, Delhi 110 016
steinke | lee | marwedel@ls12.cs.uni-dortmund.de
University of Dortmund, Dept. of Computer Science
44221 Dortmund, Germany

ABSTRACT

In this paper we address the problem of on-chip memory selection for computationally intensive applications, by proposing scratch pad memory as an alternative to cache. Area and energy for different scratch pad and cache sizes are computed using the CACTI tool while performance was evaluated using the trace results of the simulator. The target processor chosen for evaluation was AT91M40400. The results clearly establish scratchpad memory as a low power alternative in most situations with an average energy reduction of 40%. Further the average area-time reduction for the scratchpad memory was 46% of the cache memory.[1]

1. Introduction

The salient feature of portable devices is light weight and low power consumption. Applications in multimedia, video processing, speech processing, DSP applications and wireless communication require efficient memory design since on chip memory occupies more than 50% of the total chip area [1]. This will typically reduce the energy consumption of the memory unit, because less area implies reduction in the total switched capacitance. On chip caches using static RAM consume power in the range of 25% to 45% of the total chip power [2]. Recently, interest has been focussed on having on chip scratch pad memory to reduce the power and improve performance. On the other hand, they can replace caches only if they are supported by an effective compiler. Current embedded processors particularly in the area of multimedia applications and graphic controllers have on-chip scratch pad memories. In cache memory systems, the mapping of program elements is done during runtime, whereas in scratch pad memory systems this is done either by the user or automatically by the compiler using suitable algorithm.
Although prior studies into scratch pad memory behavior for embedded systems have been conducted, the impact on area have not been addressed. This paper compares cache/scratch pad area models along with their energy models. Specifically we address the following issues

1. To support comparison of memory systems we generate area models for different cache and scratchpad memory. Further, energy consumed per access for cache and scratchpad is computed for different sizes of cache and scratchpad.

2. We develop a systematic framework to evaluate the area-performance tradeoff of cache/scratch pad based systems. Experimental environment requires the use of a packing algorithm (which is a compiler support) to map the elements onto the scratchpad memory.

3. Finally, we report the performance and energy consumption for different cache and scratchpad sizes, for the various applications. We include the main memory energy consumption to study the complete system energy requirements.

The rest of the paper is organized as follows. In section 2 we explain the scratch pad memory area and energy models. In section 3, we present cache memory used in our work. Section 4 describes the methodology and the experimental setup and section 5 contains the results. In section 6 we conclude and also specify the future work.

2. Scratch pad memory

The scratch pad is a memory array with the decoding and the column circuitry logic. This model is designed keeping in view that the memory objects are mapped to the

[1]This project is supported under DST-DAAD grants project number MCS 216

scratch pad in the last stage of the compiler. The assumption here is that the scratch pad memory occupies one distinct part of the memory address space with the rest of the space occupied by main memory. Thus, we need not check for the availability of the data/instruction in the scratch pad. It reduces the comparator and the signal miss/hit acknowledging circuitry. This contributes to the energy as well as area reduction.

The scratch pad memory array cell is shown in Fig. 1(a) and the memory cell in 1(b).

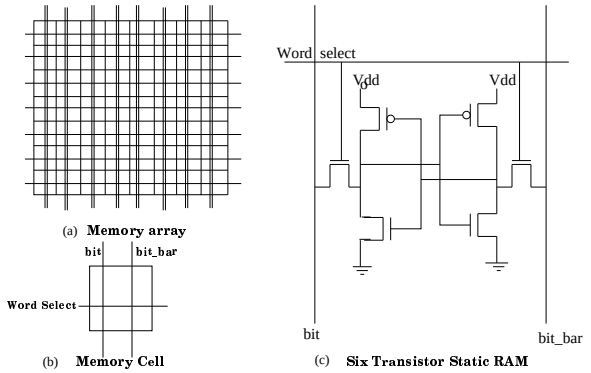

Figure 1: Scratch memory array

The 6 transistor static RAM cell is shown in Fig 1(c). The cell has one R/W port. Each cell has two bit-lines, bit and bit bar, and one word-line.The complete scratch pad organization is as shown in Fig. 2.

From the organization shown in Fig. 2, the area of the scratch pad is the sum of the area occupied by the decoder, data array and the column circuit. Let A_s be the area of the scratch pad memory.

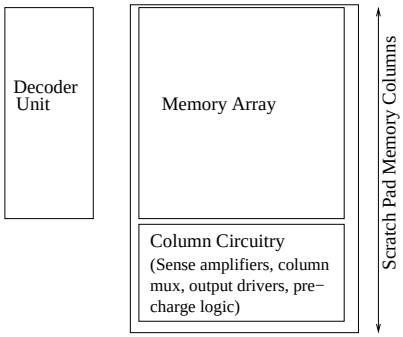

Figure 2: Scratch pad memory organization

$$A_s = A_{sde} + A_{sda} + A_{sco} + A_{spr} + A_{sse} + A_{sou} \quad (1)$$

where A_{sde}, A_{sda}, A_{sco}, A_{spr}, A_{sse} and A_{sou} is the area of the data decoder, data array area, column multiplexer, pre-charge, data sense amplifiers and the output driver units respectively.
The scratch pad memory energy consumption can be estimated from the energy consumption of its components i.e. decoder $E_{decoder}$ and memory columns E_{memcol}.

$$E_{scratchpad} = E_{decoder} + E_{memcol} \quad (2)$$

Energy in the memory array consists of the energy consumed in the sense amplifiers, column multiplexers, the output driver circuitry, and the memory cells due to the word-line, pre-charge circuit and the bit line circuitry. The major energy consumption is due to the memory array unit. The procedure followed in the CACTI tool to estimate the energy consumption is to first compute the capacitances for each unit. Then, energy is estimated. As an example we only describe the energy computation for the memory array. Similar analysis is performed for the decoder circuitry also, taking into account the various switching activity at the inputs of each stage.

Let us consider the energy dissipation E_{memcol}. It consists of the energy dissipated in the memory cell. Thus

$$E_{memcol} = C_{memcol} * V_{dd}^2 * P_{0->1} \quad (3)$$

C_{memcol} in equation (3) is the capacitance of the memory array unit. $P_{0->1}$ is taken as 0.5 is the probability of a bit toggle.

$$C_{memcol} = ncols * (C_{pre} + C_{readwrite}) \quad (4)$$

C_{memcol} is computed from equation (4). It is the sum of the capacitances due to pre-charge and read access to the scratch pad memory. C_{pre} is the effective load capacitance of the bit-lines during pre-charging and $C_{readwrite}$ is the effective load capacitance of the cell read/write. $ncols$ is the number of columns in the memory.

In the preparation for an access, bit-lines are pre-charged and during actual read/write, one side of the bit lines are pulled down. Energy is therefore dissipated in the bit-lines due to pre-charging and the read/write access. When the scratch pad memory is accessed, the address decoder first decodes the address bits to find the desired row. The transition in the address bits causes the charging and discharging of capacitances in the decoder path. This brings about energy dissipation in the decoder path. The transition in the last stage, that is the word-line driver stage triggers the

switching in the word-line. Regardless of how many address bits change, only two word-lines among all will be switched. One will be logic 0 and the other will be logic 1. The equations are derived based on [4].

$$E_{sptotal} = SP_{access} * E_{scratchpad} \quad (5)$$

where $E_{sptotal}$ is the total energy spent in the scratch pad memory. In case of a scratch pad as a contrast to cache we do not have events due to write miss and read miss. The only possible case that holds good is the read or write access. SP_{access} is the number of accesses to the scratch pad memory. $E_{scratchpad}$ is the energy per access obtained from our analytical scratch pad model.

3. Cache memory

Caches are mainly used to exploit the temporal and spatial locality of memory accesses. The basic organization of the cache is taken from [4] and is shown in Fig. 3.

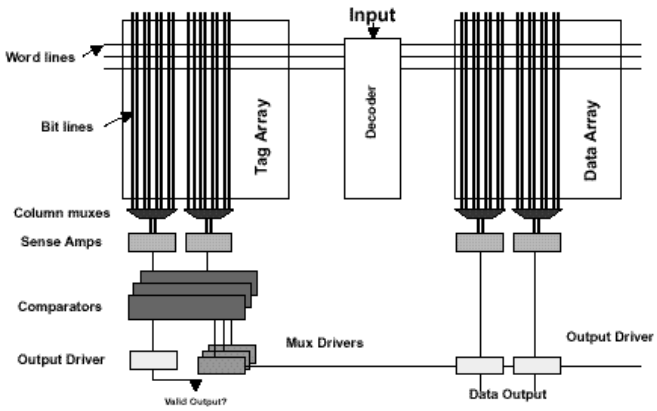

Figure 3: Cache Memory organization[4]

The area model that we use in our work is based on the transistor count in the circuitry. All transistor counts are computed from the designs of circuits.

From the organization shown in Fig. 3, the area of the cache (Ac) is the sum of the area occupied by the tag array (A_{tag}) and data array (A_{data}).

$$A_c = A_{tag} + A_{data} \quad (6)$$

A_{tag} and A_{data} is computed using the area of its components.

$$A_{tag} = A_{dt} + A_{ta} + A_{co} + A_{pr} + A_{se} + A_{com} + A_{mu} \quad (7)$$

where A_{dt}, A_{ta}, A_{co}, A_{pr}, A_{se}, A_{com} and A_{mu} is the area of the tag decoder unit, tag array, column multiplexer, the pre-charge, sense amplifiers, tag comparators and multiplexer driver units respectively.

$$A_{data} = A_{de} + A_{da} + A_{col} + A_{pre} + A_{sen} + A_{out} \quad (8)$$

where A_{de}, A_{da}, A_{col}, A_{pre}, A_{sen}, A_{out} is the area of the data decoder unit, data array, column multiplexer, precharge, data sense amplifiers and the output driver units respectively. The estimation of power can be done at different levels, from the transistor level to the architectural level [6]. In CACTI [4], transistor level power estimation is done. The energy consumption per access in a cache is the sum of energy consumptions of all the components identified above. The analysis is similar to that described for the scratch pad memory.

4. Overview of our methodology

Clock cycle estimation is based on the ARMulator trace output for cache or scratch pad memory. This is assumed to directly reflect performance i.e. the larger the number of clock cycles the lower the performance. This is under the assumption that the change in the on-chip memory configuration (cache/scratch pad memory and its size) does not change the clock period. This assumption though restrictive does not affect our results. This is because we always compare the same size cache with scratch pad memory and the delay of cache implemented with the same technology will always be higher. Thus the performance improvement predicted for scratch pad can only increase if both effect the clock period. The identification and assignment of critical data structures to scratch pad was based on a packing algorithm briefly described in 4.3.

4.1 Scratch pad memory accesses

From the trace file it is possible to do the performance estimation. As the scratch pad is assumed to occupy part of the total memory address space, from the address values obtained by the trace analyzer, the access is classified as going to scratch pad or memory and an appropriate latency is added to the overall program delay. One cycle is assumed if it is a scratch pad read or write access. If it is a main memory 16 bit access then we take it as one cycle plus 1 wait state (refer to Tabel 1). If it is a main memory 32 bit access then, we consider it as one cycle plus 3 wait states. The total time in number of clock cycles is used to determine the performance. The scratch pad energy consumption is the number of accesses multiplied by the energy per access as described in equation 5.

4.2 Cache memory accesses

From the trace file it is possible to obtain the number of cache read hits, read misses, write hits and write misses.

Access	Number of cycles
Cache	Using Table 2
Scratch pad	1 cycle
Main Memory 16 bit	1 cycle + 1 wait state
Main Memory 32 bit	1 cycle + 3 wait states

Table 1: Memory access cycles

From this data we compute the number of accesses to cache based on Table 2, where the number of cycles required for each type of access is listed in Table 1. The cache is a write through cache. There are four cases of cache access that we consider in our model.

- **Cache read hit** : When the CPU requires some data, the tag array of the cache is accessed. If there is a cache read hit, then data is read from the cache. No write to the cache is done, and main memory is not accessed for a read or write.
- **Cache read miss** : When there is a cache read miss, it implies that the data is not in the cache, and the line has to be brought from main memory to cache. In this case we have a cache read operation, followed by L words to be written in the cache, where L is the line size. Hence there will be a main memory read event of size L with no main memory write.
- **Cache write hit** : If there is a cache write hit, we have a cache write, followed by a main memory write.
- **Cache write miss** : In case of a cache write miss, a cache tag read (to establish the miss) is followed by the main memory write. There is no cache update in this case.

Access type	Ca_{read}	Ca_{write}	Mm_{read}	Mm_{write}
Read hit	1	0	0	0
Read miss	1	L	L	0
Write hit	0	1	0	1
Write miss	1	0	0	1

Table 2: Cache memory interaction model

Using this model we derive the cache energy equation as

$$E_{cache} = (N_{c-read} + N_{c-write}) * E \qquad (9)$$

Where E_{cache} is the energy spent in cache. N_{c-read} is the number of cache read accesses and $N_{c-write}$ is the number of cache write accesses. Energy E is computed like in equation (3), taking the appropriate load and the number of cycles into consideration. In the trace analyzer we model the cache as described above and use it in our performance and energy estimations.

4.3 Experimental setup and flow diagram

In this subsection, we explain the experimental setup and flow diagram used in our work to compare on-chip scratch pad memory with cache memory. We use the AT91M 40400 as our target architecture. The AT91M 40400 is a member of the ATMEL AT91 16/32 bit microcontroller family based on the ARM7TDMI embedded processor. This processor is a high performance RISC with a very low power consumption. It has an on-chip scratch pad memory of 4 KBytes. The ARM7TDMI comes with a 32 bit data path and two instruction sets.

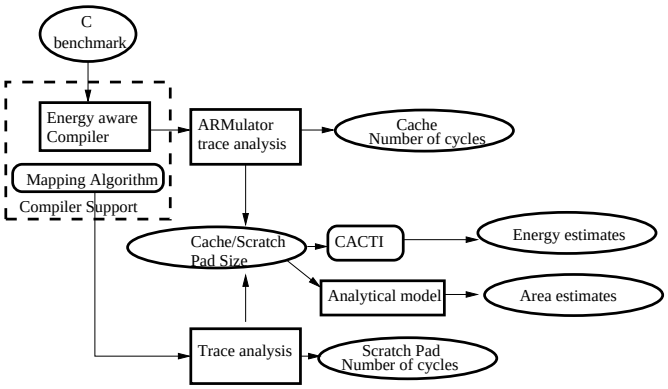

Figure 4: Experimental flow diagram

Fig. 4 shows the flow diagram. The energy aware (encc) compiler [7] generates the code for the ARM7 core. It is a research compiler used for exploring the design and new optimization techniques. The input to this compiler is an application benchmark written in C. As a post pass option, encc uses a special packing algorithm, known as the knapsack algorithm [5], for assigning code and data blocks to the scratch pad memory. This algorithm identifies the frequently referred data and instruction blocks and maps to the scratch pad memory address space. The cost of additional jumps introduced due to mapping consecutive blocks to scratch pad and main memory is accounted for by the algorithm. The result is that blocks of instructions and data which are frequently accessed, and are likely to generate maximum energy savings, are assigned to the scratch pad. The output of the compiler is a binary ARM code which can be simulated by the ARMulator to produce a trace file. For on-chip cache configuration, the ARMulator accepts the cache size as parameter and generates the performance as the number of cycles. The predicted area and energy is based on the CACTI [4] model for 0.5 μm technology. The models themselves are described in sections 2 and 3.

5. Results

To demonstrate the merits of using on-chip scratch pad memory and on-chip caches, we conducted a series of experiments for both of these configurations. The trace analysis

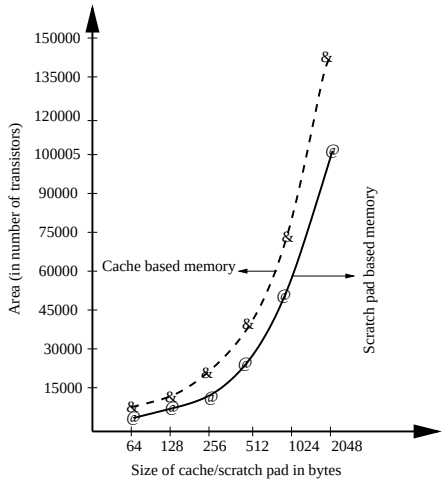

Figure 5: Comparison of cache and scratch pad memory area

for the scratch pad and the cache was done in the design flow after the compilation phase. We use a 2-way set associative cache configuration for comparison. The area is represented in terms of number of transistors. These are obtained from the cache and scratch pad organization. Fig. 5 shows the comparison of area of the cache and scratch pad memory for varying sizes. We find that on an average the area occupied by the scratch pad is less than the cache memory by 34%.

Table 3 gives the area/performance tradeoff. Column 1 is the size of scratch pad or cache in bytes. Columns 2 and 3 are the cache and scratch pad area in transistors. Columns 4 and 5 are the number of CPU cycles in 1000s for cache- and scratch-pad based memory systems, respectively. Column 6 gives the area reduction due to replacing a cache by a scratch pad memory while column 7 corresponds to the reduction in the number of cycles. Column 8 gives the the improvement of the area-time product AT (assuming constant cycle times). The area time product AT is computed using

$$AT = (A_s * N_s)/(A_c * N_c) \quad (10)$$

The average area, time, and AT product reductions are 34%, 18% and 46%, respectively for this example. Our cycle count considerations in performance evaluation are based on the static RAM chips found on an ATMEL evaluation board. To compare the energy, we need to account for the energy consumption of the main memory as well. The energy required per access by various devices is listed in table 4. The cache and scratch pad values for size 2048 bytes were obtained from models in section 2 and 3, the main memory values were obtained from actual measurements on the ATMEL board [5].
Thus we take the main memory energy, along with the on-chip memory energy consumption into account. Fig.6 shows the energy consumed for biquad, matrixmult and quicksort examples for both cache and scratch pad. In all the cases we observe that scratch pad consumes less energy for the

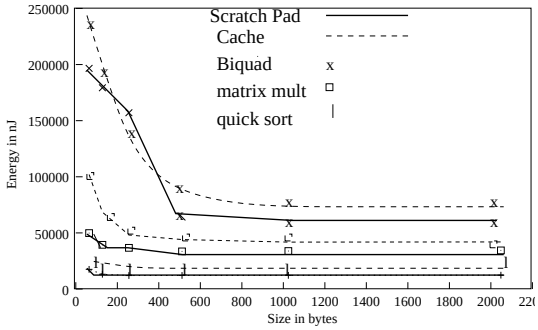

Figure 6: Energy consumed by the memory system

Cache per access (2 kbytes)	4.57 nJ
Scratch pad per access (2 kbytes)	1.53 nJ
Main memory read access, 2 bytes	24.00 nJ
Main memory read access, 4 bytes	49.30 nJ
Main memory write access, 4 bytes	41.10 nJ

Table 4: Energy per access for various devices

same size of cache, except for quicksort with cache size of 256 bytes. On an average we found energy consumption to be reduced by 40% using scratch pad memory.

6. Conclusion and future work

In this paper we have presented an approach for selection of on-chip memory configurations. The paper presents a comprehensive methodology for computing area, energy and performance for various sizes of cache and scratch pad memories. Results indicate that, scratch-pad based compile-time memory outperform cache-based run-time memory on almost all counts. We observe that the area-time product (AT) can be reduced by 46% (average) by replacing cache by the scratch pad memory. We found that, for most applications and memory configurations, the total energy consumption of scratch pad based memory systems is less than that of cache-based systems. The average reduction was 40% in the application considered. Since memory bandwidth and on-chip memory capacity are limiting factors for many applications, DRAM based memory comparisons should be studied. The cache and scratch pad energy models need to be validated by real measurements.

7. REFERENCES

[1] Doris Keitel-Sculz and Norbert Wehn., *Embedded DRAM Development Technology, Physical Design, and Application Issues*, IEEE Design and Test of Computers, Vol 18 Number 3, Page 7-15, May/June 2001.

[2] Preeti Ranjan Panda, Nikhil Dutt, Alexandru Nicolau : *Memory issues in embedded systems on-chip*

Size bytes	Area Cache A_c	Area Scratchpad A_s	CPU cycles cache N_c	CPU cycles Scratchpad N_s	Area reduction	Time reduction	Area-time Product
64	6744	4032	481.9	347.5	0.40	0.28	0.44
128	11238	7104	302.4	239.9	0.37	0.21	0.51
256	21586	14306	264.0	237.9	0.34	0.10	0.55
512	38630	26722	242.6	237.9	0.31	0.10	0.61
1024	74680	53444	241.7	192.0	0.28	0.21	0.55
2048	142224	102852	241.5	192.0	0.28	0.20	0.57
Average					0.33	0.18	0.54

Table 3: Area/Performance ratios for bubble-sort

- *Optimisations and exploration*, Kluwer Academic Publishers, 1999.

[3] V. Zivojnovic, J. Velarde, and C. Schlager : *DSPStone : A DSP-oriented benchmarking methodology*, In Proceedings of the 5th International Conference on Signal Processing Applications and Technology, October 1994.

[4] S Wilton and Norm Jouppi : *Cacti : An enhanced access and cycle time model*, IEEE Journal of Solid State Circuits, May 1996.

[5] Rajeshwari Banakar, S Steinke, B S Lee, M Balakrishnan and P Marwedel, *Comparison of cache and scratch pad based memory system with respect to performance, area and energy consumption*, Technical Report 762, University of Dortmund, Sep 2001.

[6] Rajeshwari M Banakar, Ranjan Bose, M Balakrishnan : *Low power design - Abstraction levels and RTL design techniques*, VLSI test and design Workshop, VDAT 2001 Bangalore, Aug 2001

[7] *ls12-www.cs.uni-dortmund.de/research/encc*

[8] Luca Benini, Alberto Macii, Enrico Macii, Massino Poncino : *Synthesis of application specific memories for power optimisation in embedded systems*, DAC 2000 Los Angeles, California, pp 300-303.

[9] J Kin, M Gupta and WH Mangonie-Smith : *The filter cache: An energy efficient memory structure*, IEEE Micro-30 December 1997.

[10] T Ishihara and H Yasuura : *A power reduction technique with object code merging for application specific embedded processors*, Proceedings of Design Automation and Testing, Europe Conference (DATE 2000), March 2000.