EECS 507 Presentation

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

Content

- Introduction
- Scratch pad memory
- Cache memory
- Methodology
- Results
- Conclusion

INTRODUCTION

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

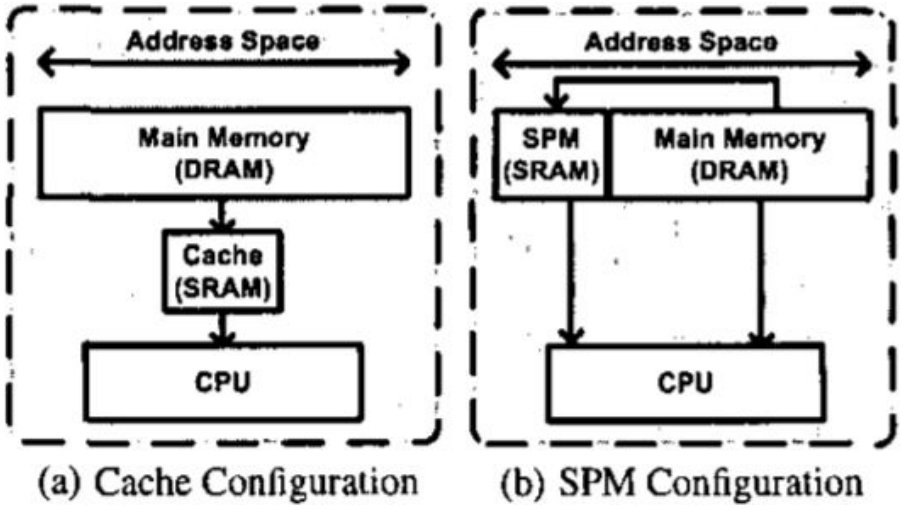
- Modern embedded device is light weight and low power consumption
  - Chip caches using 25% to 45% total chip power
  - Need a more efficient compiler

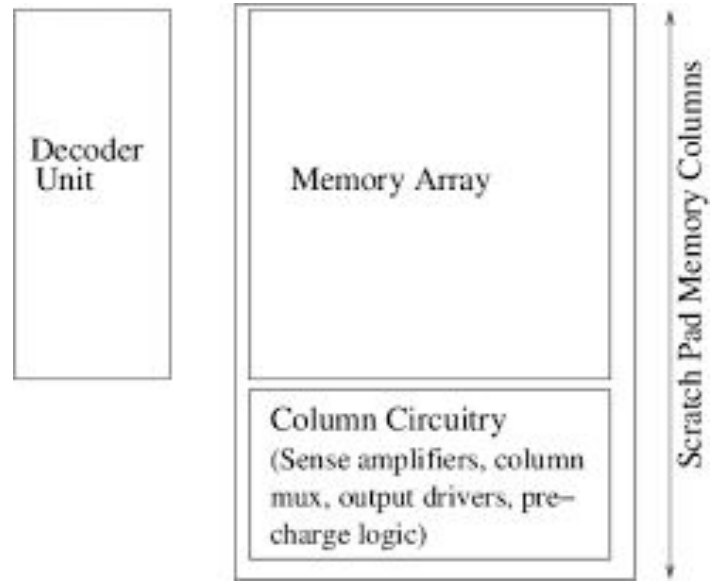
Main Contribution:

1. Establishing area model to evaluate and compare different caches and scratch pad memory, further different size of caches and scratch pad memory
2. Using a systematic framework to evaluate the area-performance tradeoff of cache and scratch pad based system
3. Report and analysis the performance and energy consumption for different caches and scratch pad memory

Scratch Pad Memory

- Memory array with the decoding and the column circuitry logic
- Scratch pad memory (benefits)





Scratch Pad Memory

- Major energy consumption is memory array unit

Energy dissipation formula: $E_{memcol} = C_{memcol} * V_{dd}^2 \bullet P_{0->1}$

Capacitance of the memory array unit: $C_{memcol} = ncols * (C_{pre} + C_{readwrite})$

Total energy spent in the scratch pad memory:

$$E_{sptotal} = SP_{access} * E_{scratchpad}$$

SPaccess is the number of accesses to the scratch pad memory. $E_{scratchpad}$ is the energy per access obtained from our analytical scratch pad model.

Cache Memory

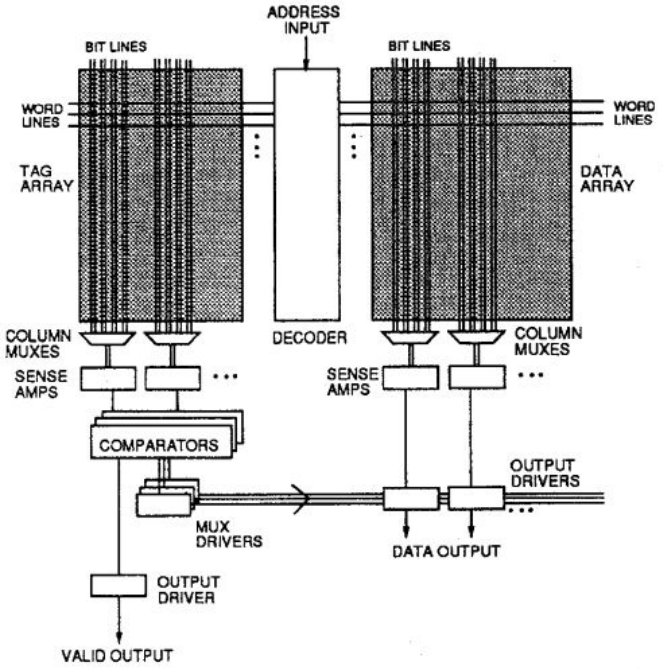
Area Model

- base on the transistor count in the circuitry

$$A_{cache} = A_{tag} + A_{data}$$

A_{tag} means area of tag array

A_{data} means area of tag array



Cache Memory Organization

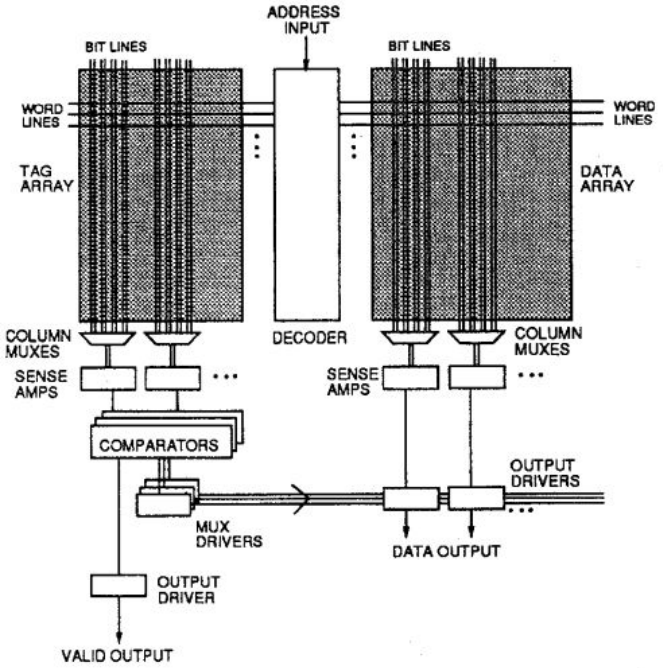
Cache Memory

$$A_{tag} = A_{dt} + A_{ta} + A_{co} + A_{pr} + A_{se} + A_{com} + A_{mu}$$

Above is the area of

- tag decoder unit
- tag array
- column multiplexer
- pre-charge
- sense amplifiers
- tag comparators
- multiplexer driver units

Respectively



Cache Memory Organization

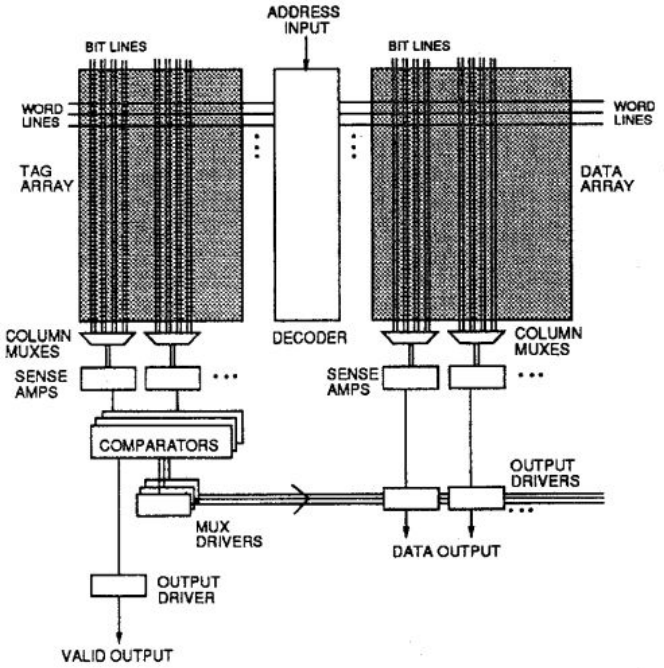
Cache Memory

$$A_{data} = A_{de} + A_{da} + A_{col} + A_{pre} + A_{sen} + A_{out}$$

Above is the area of

- data decoder unit
- data array
- column multiplexer
- pre-charge
- data sense amplifiers
- output driver units

Respectively



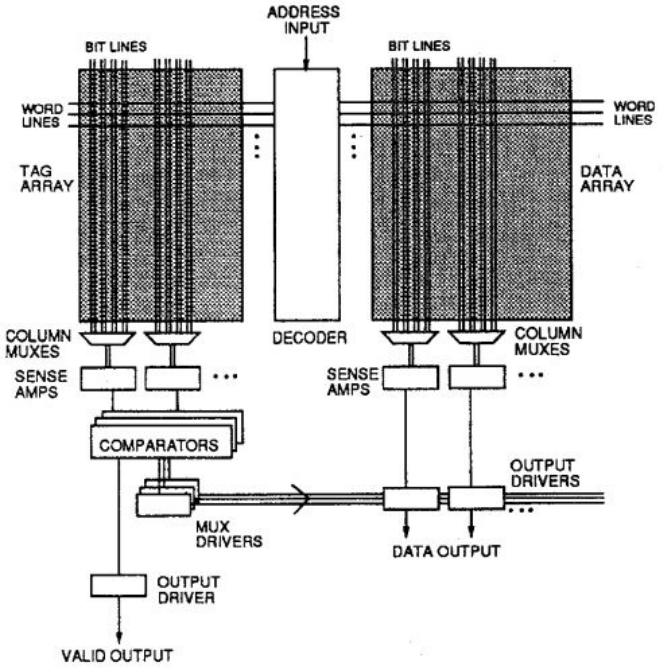
Cache Memory Organization

Cache Memory

Power Model

- energy consumption for each access in a cache is the sum of energy consumptions of all the components identified in area model and is obtained from CACTI

$$E_{cache} = E_{tag} + E_{data}$$



Cache Memory Organization

METHODOLOGY

- Quantitatively evaluate usage of Scratchpad memory vs Cache for Embedded applications.
- Evaluation Criteria:
  - Performance
  - Energy
  - Area
- Evaluation Platforms used:
  - ARMulator - Performance
  - CACTI - Energy and Area

PERFORMANCE MODEL

- ARMulator instruction set simulator is used that generates trace output for Cache and SPM.
- The number of clock cycles per access determines the performance of Cache or SPM.

SCRATCH PAD MEMORY ACCESS

- Addresses are classified as going to SPM or memory based on outputs of trace analyzer.
- SPM read and write access time is assumed to be one clock cycle.

Access	Number of cycles
SPM	1 cycle
Main Memory 16 bit	1 cycle + 1 wait state
Main Memory 32 bit	1 cycle + 3 wait states

CACHE MEMORY ACCESS

- Trace file give the number of cache read hits, read misses, write hits, write misses.
- The cache model used in the paper is a writethrough.

Cases of Cache access considered in the model:

- **Cache read hit:**
  - Data read from Cache
  - No write to Cache
  - No memory access

- **Cache read miss:**
  - Main memory accessed
  - L words written to Cache

- **Cache write hit:**
  - Cache write
  - Main memory write

- **Cache write miss:**
  - Main memory write
  - No cache update

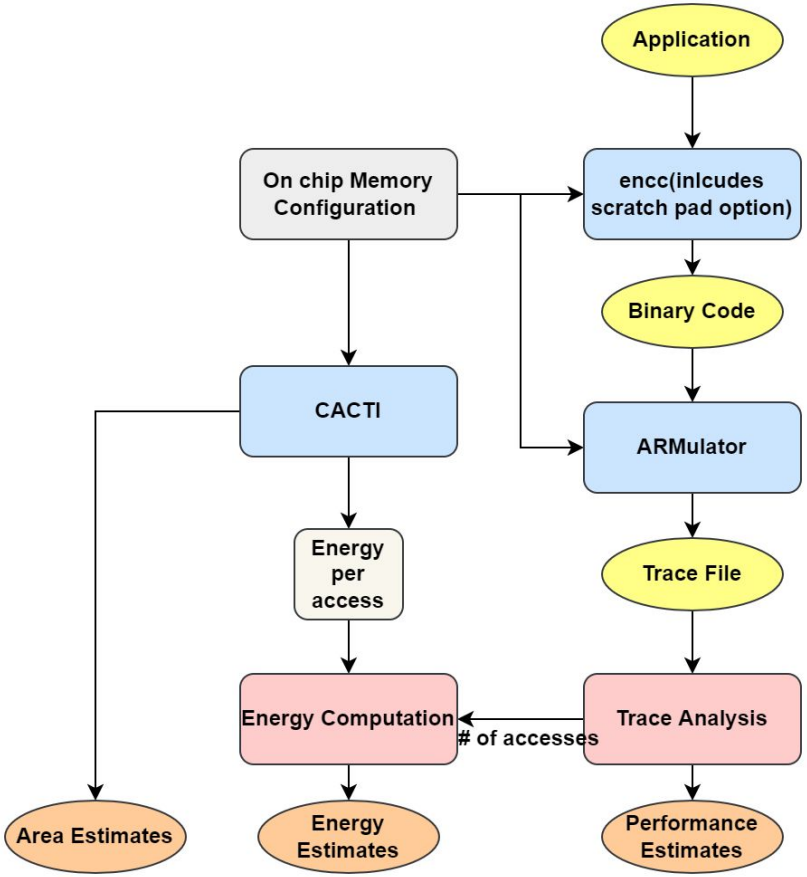
Access type	Cache read	Cache write	Mem read	Mem write
Read hit	1	0	0	0
Read miss	1	L	L	0
Write hit	0	1	0	1
Write miss	1	0	0	1

Cache Energy Eq:

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

EXPERIMENTAL SETUP AND FLOW

- Target architecture: AT91M40400 is a member of the ATMEL AT91 16/32 bit microcontroller family based on the ARM7TDMI embedded processor.
- 4KB on chip SPM.
- Encc compiler generates the binary code for ARM7 which is simulated by ARMulator to generate trace file.
- Knapsack algorithm is used for assigning code and data blocks for SPM.
- CACTI model is used to predict area and energy for Cache and SPM



Selection of memory objects for Scratch Pad

- A set of functions, basic blocks and/or variables is assigned to the scratch pad on a static basis.
- Algorithm in the compiler first identifies and evaluates frequent program and data memory objects.
- Program memory objects - Functions and basic blocks are identified to be assigned to SPM.
- Energy consumption is computed by multiplying energy consumption of a single execution with number of executions of the memory objects, $E = n * E_{instr\_fetch}$
- Data memory objects - Along with program, variables can also be assigned to SPM.
- The number of accesses of the variable is the number of static references in the block times number of block executions, $E = n * E_{data\_word}$
- The best set of memory objects which fit into the scratch pad and save the highest amount of energy now has to be identified.
- Maximizing the total gain is a problem that can be modeled as a knapsack problem.

Performance Results

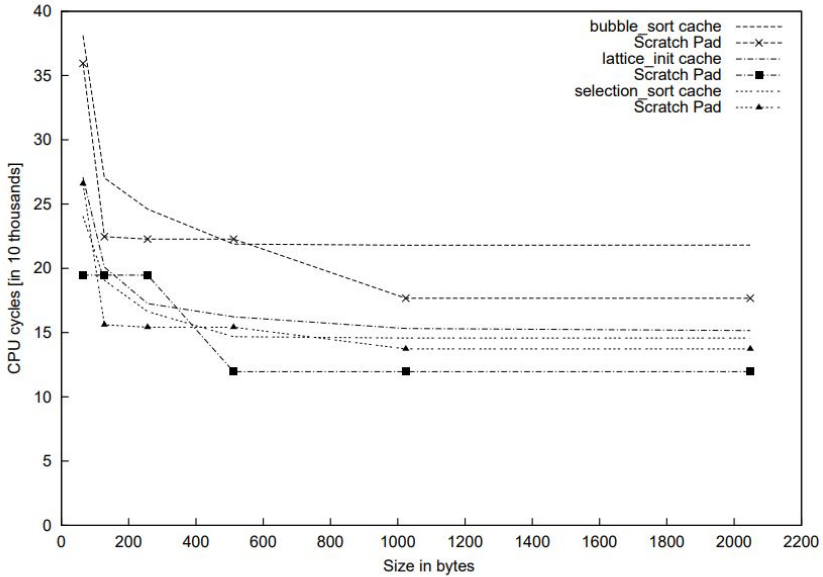


Figure 11: CPU cycles for cache and scratch pad memory

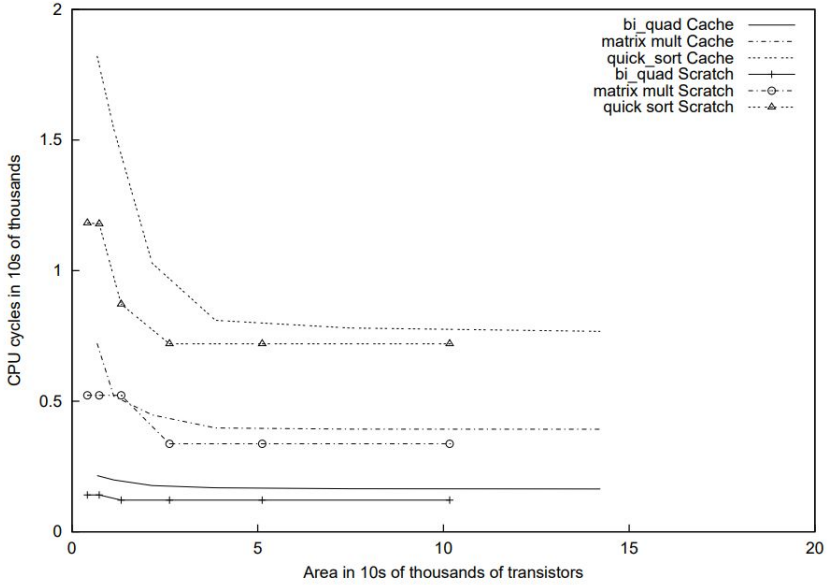
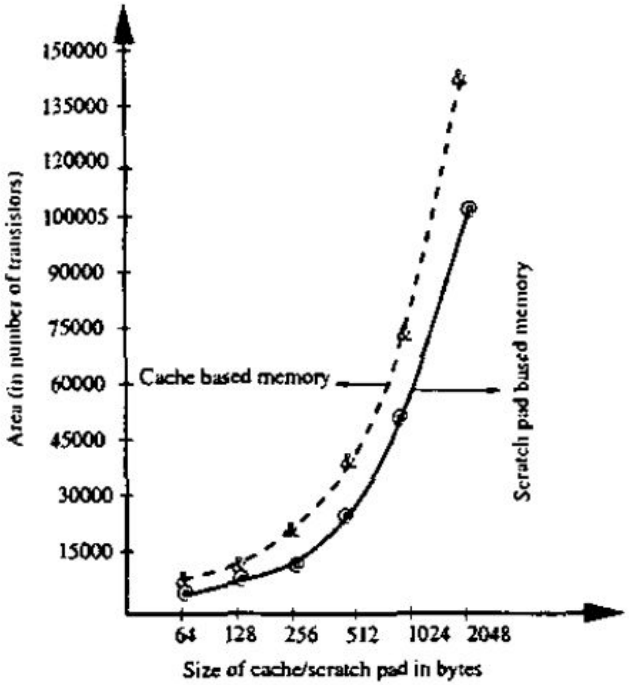


Figure 12: Area vs Performance biquad, matrix mult, quick sort

Area-Time Results



Parameters -

- 2-way set associative cache
- Performance measured by CPU cycles
- Data for bubble-sort

Average reduction in Scratchpad (SPM) compared to Caches -

- Area - 34%
- Time - 18%
- Area-time product - 46%

$$AT = (A_s * Ns)/(A_c * Nc)$$

Energy Results

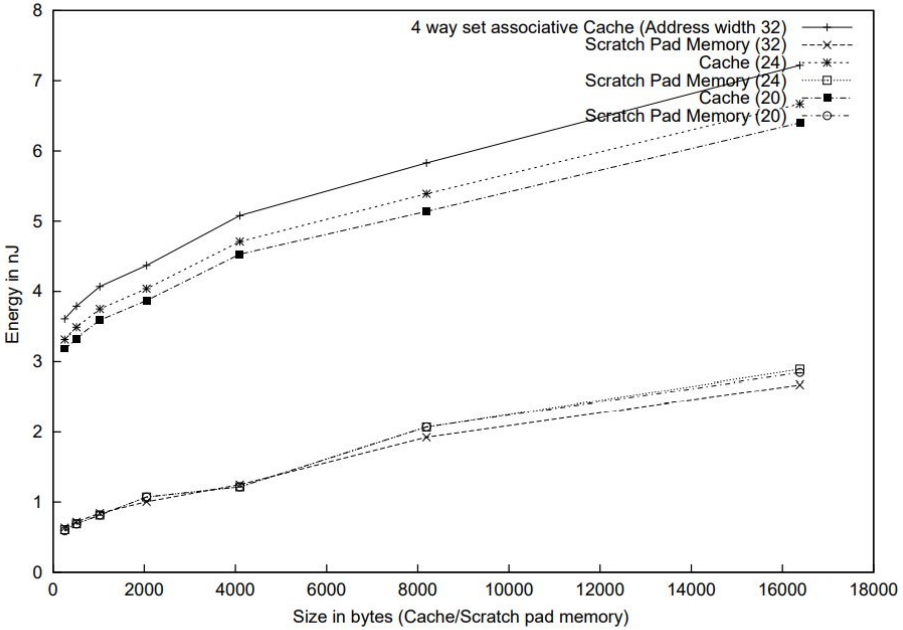


Figure 15: Scratch pad memory and cache energy comparison

Energy per access -

- Cache (2KB) - 4.57 nJ
- SPM (2KB) - 1.53 nJ

Average reduction - 40%

Conclusion & Future Work

- For small size (less than 600B), ambiguous results
- For large size, SPM static allocation > cache dynamic (in terms of area, energy, time)

- DRAM main memory should be studied
- Needs real measurements
- Clock period of SPM vs cache (currently, CPU cycles)
- Cost/performance direct mapped caches

Unanswered Questions

- Results based on few algorithms. Use benchmarks? Real world more complex
- Static memory allocation. Possible/Optimal for all programs? If not, do programmers allocate? Binary compatibility?
- Paper published in 2002. Now, multiprocessors. Common SPM? No coherence issue
- Write through cache. What about write back?
- No OS? No VM? Virtually or physically indexed SPM? Context switching?
- Mix of both cache & SPM

Thank you!