An optimization technique to improve power consumption of Embedded System

[1]G.INDRA REDDY
[1]assistant professor,
[1]electrical & electronics department,
[1]guru nanak instittutions technical campus, hyderabad, india

Abstract: **Embedded systems with predetermined applications can attain further up gradation by the addition of fast and energy efficient scratchpad memory (SPM) on chip and moving frequent accesses code and/or data from main memory to SPM. Many microprocessors spends majority of its time waiting for the data to arrive from slow memory devices connected to it. This is termed as memory wall problem. The main aim of this paper is to reduce memory wall problem by increasing the speed of the system not only by processor speed but also by the memory speed. Also this paper proposes a two-stage method which works on improving the efficiency of the memory by focusing on both on chip and off chip memory, thereby reducing the memory wall problem. Experimental results reveal 1) allocating a dedicated portion of the on-chip processor memory as SPM 2) studying the combined benefit of cache and SPM for both single and multiple processor 3) Using external cache alone with on chip SPM and cache which can further minimize the energy consumption of the embedded systems. Simulation result for our method reduces the energy consumption of the system by 3%.Energy gain can be obtained by using a external cache alone with on chip cache and SPM.**

Index Terms—Memory Wall Problem, Embedded System,Cache, Scratch Pad Memory.

I.INTRODUCTION

Embedded system is a portable device .The main design consideration of this type of device is light weight and reduced power consumption. Two types of processor architecture are CISC (Complex Instruction Set Computers) and RISC (Reduced Instruction Set Computers).CISC uses bigger program hence occupies much more space and in turn consume more capital than RISC. Embedded system which mainly focuses on low energy and light weighted product does not prefer CISC for their development. Hence we go for RISC which executes multiple commands in single operation. Multiple operations in single instruction lead to fast memory access which in turn lead to the need of fast memory device as fetch and decode will take place faster in RISC. So the usage of RISC as a processor on embedded system is highly preferred.

Until now cache is considered to be the memory which is mainly used to provide the transparency between the processor and the memory. In fact caches are still used in many general purpose processors. Many studies have recently proposed Scratch Pad Memory (SPM) which is a software controlled memory .SPM consists of only data array and memory decoder logic where as cache consists of data array, memory decoder logic along with tag memory which makes it both larger in size and power consumption than SPM, hence SPM is efficient than cache.SPM require an explicit support such as strong programming or some compiler directed tool for their proper utilization , a strong programming logic can make efficient use of this memory. SPM consume 40% less energy and 34% less space than cache [1]. Execution time can be accurately predicted in SPM. Programmer can determine what part of code or data should be placed into the SPM, it uses special high speed memory circuit to hold small item of data for rapid retrieval. SPM simplifies caching logic and helps system work without main memory contention especially in cache of MPSOC (multiprocessor system on chip).

Traditionally software for embedded system was developed using assembly language but today high level language is used for the programming, as embedded system need compactness, hence replacing the hardware with software yields low hardware consumption there by size will be reduced .In turn one can make maximum utilization of system with low power consumption, hence making proper utilization of software will produce a high speed and energy efficient device.

Embedded processor such as ARM11 [2], Cold fire [3] and CELL [4] uses on chip scratch pad memory. Programming and compiler can move the frequently used code or data object to SPM thereby reducing the power and improving the efficiency of the system. Sony Play station 2 is an example which uses 16 KB Scratch Pad memory. It is a gaming device which uses the steady stream of graphics and audio which demands high speed memory.

The paper is organized as follows Section 2 describe related research on scratch pad memory and cache memory section 3 deals with the methodology used in the paper section 4 present the experimental setting and result .An overview on how system works by combining both approaches is discussed in detail.

II.RELATED WORK

Using Scratch pad memory and cache together will improve the efficiency of the embedded system. Many research works focus to improve the efficiency of embedded system using both hardware and software memory at the same time. When both the memories are used together along with some optimization techniques will improve the efficiency of the system. Optimization mainly aims at improving the efficiency of the system. To achieve the competence main focus of the research goes to on chip memory. Software controlled memory plays an important role in real time system with hard deadline as it allow the programmer to accurately predict the time for processing of the data code or the object code. Panda et al. [5] demonstrated the benefits of using both cache and SPM in embedded systems. This advance panels application data variables to dislodge address spaces occupied by SPM and main memory, making it feasible to obtain the collective benefits of both methods and to develop the performance of dynamic binary transformation in resource-constrained embedded systems Ishitobi et al. [6] revealed the benefits of improving cache deeds by suitably rearranging the order of program objects (code and data) in the memory space into non-cacheable, cacheable, and SPM regions. On the other hand, their approach chairs program objects in compacted contiguous position. This is an needless constraint since the cache mapping rule is governed solely by memory position, which evaluate whether two series items battle for the same cache block. Thus, the universal approach is to permit gaps in between the program objects in the main memory which can further relieve cache conflicts, methods of allocation they have used locality optimization technique to achieve the efficiency. Another technique used for placing data and code for SPM and cache memory is discussed in [12].The procedure find the code layout for cacheable region and scratch pad memory region this minimize the energy consumption for embedded system, the code layout works as follows primarily hardware reliant parameter such as energy consumed and clock cycle require for memory contact is found out by using net list of target processor. Then instruction and data address trace for target program is established out using instruction set simulator (ISS).Code placement algorithm finds optimal code layout using formerly obtained hardware and software parameter. In dynamic data scratchpad Memory Management [7].SPM has a clearly managed run time application or a committed SPM administrator, which is part of run time environment. It uses demand paging practice which is similar to virtual memory. Loading of data objects are made by triggering of a function which is the job of SPMM (Scratch Pad Memory Management) .SPMM management calls are inserted before and after the function call, this SMMM call manages the page table and carry out virtual to physical translation, data can be placed into SPM without using SPMM by the method of 0-1 Knapsack problem. Manish verma[11] obtain a trace in which the program blocks are brought into the memory during the execution for different size of cache and is observed by using control flow graph(CFG) , by using weighted CFG layout and execution trace energy subsystem was premeditated. And energy value for system using one word SPM and two word SPM are calculated, then energy value for loop cache of one word and two word are designed. This technique removes the cache miss and improves the energy efficiency of the method. Tanja Van Achteren and Francky Catthoor [13] uses the replacement policy of cache , which considers last access history to replaced or overwrites the cache by the approach which checks at compile time the fresh data which has to be positioned in cache and have future reuse to improve power cost , given by a vector/matrix abstraction technique. M Kandemir [12] obtains the information which either adapts application to existing memory hierarchy or come up with new memory hierarchy In [14] a strategy is used to minimize the total execution time of embedded system by carefully partitioning the variable used in the application among off chip and Scratch Pad Memory.

III. METHODOLOGY

This work adopt the Harvard architecture shown in Fig. 1, Considering scratchpad memory locates on the same level as the set-associative level-1 (L1) instruction cache. While a agenda consists of a quantity of data and code objects, this revision only consider the plan of code objects. A code object is loaded to SPM or main memory depending on the code layout design. If a code object is loaded to the main memory, it occupies a number of adjacent memory blocks. Believing that the size of a memory block equals the size of SPM and the cache block size is smaller than the main memory and SPM size. The proposed system , design a architecture which consists of On chip SPM and cache alone with external cache which avoid all the cache misses. This external cache is placed between the main memory and On chip. Any miss in SPM or internal cache will direct the search to external cache instead of going to the main memory which will save the energy and advance the performance. Scratch Pad Memory is simple SRAM memory placed on chip along with the processor. SPM consumes less CPU cycle and energy, unlike the main memory the size of the scratchpad memory is limited to be a fraction of the total application size.

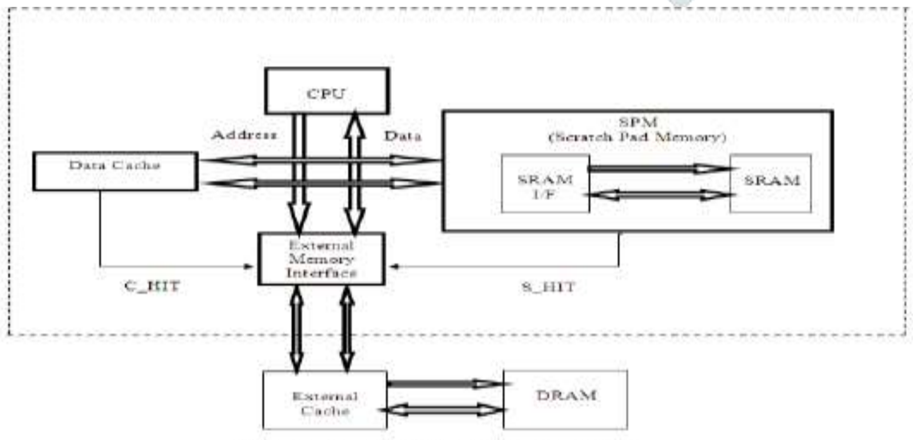


Fig1: Architecture Diagram

A. SCRATCH PAD MEMORY Software managed memory such as scratch pad memories are important particularly in case of image and video processing application where heavy use of multi dimensional array is implemented. As these applications need large area for its storage, scratch pad architectures are used, which results in large saving in energy and improvement in the efficiency of the system. Scratch pad memory based system uses, the programmer or the compiler who are responsible for scheduling the data transfer between the SPM and the off chip memory. Effective scheduling of memory will improve both the efficiency and performance of the application.

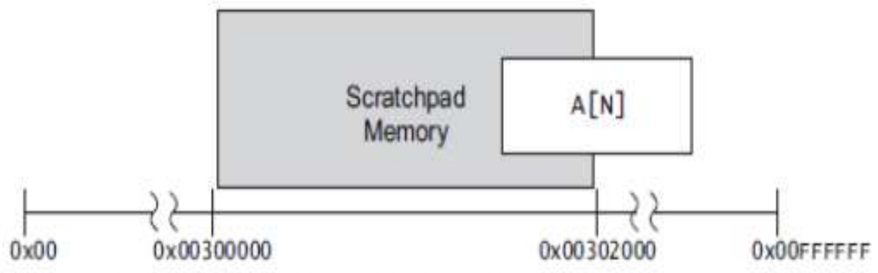


Fig 2: Processor Address Space Containing a Scratchpad Memory

In most of the embedded systems scratch pad memory occupy small portion of the memory address space [1]. Figure2 shows a setup in which the scratchpad occupies a 4k address region ([0x00300000, 0x00302000]) from the processor's address space ([0x00000000, 0x00FFFFFF]).Any access to the 4k address region is translated to a scratchpad access, whereas any other address access is mapped to the main memory. Scratchpad memories, consisting of data memory and address decoding circuitry it require less on chip area. However unlike caches, scratchpads require complex program analysis and explicit support from the compiler. B. CACHE MEMORY Caches are mainly used to avoid the spatial and temporal locality of memory access. If one ignores the SPM then cache is the name given to the highest level of memory hierarchy once the address leaves the processor .The word cache is generally used where buffering or reusing of data is employed such as file cache, name cache etc. Cache is used to reduce the average time to access memory Three different type of cache used are • Instruction cache-It is mainly used to speed up the instruction fetch. • Data fetch- It is mainly used to speed up the LOAD and STORE operation • Translation Look Up Header- Used to speed up the virtual to physical address translation Data cache is organized as hierarchy; data are transferred in fixed size of block called cache line. Cache entry structure consists of tag, flag bit and data block. Tags are used to translate from cache address to unique CPU address. Flag bit indicates whether or not flag bit are loaded with valid data. Data block contains the original data. Cache set replacement policy are used to find which block has to be replaced first, according to the replacement policy overwriting of the cache data is performed. The energy consumption in the cache is the sum of the entire components identified above. From the complex diagram of cache compared to the scratch pad memory it is clear that the cache consumes high space and energy than SPM. The goal of the cache aware scratchpad allocation (CASA) problem is to minimize the energy consumption of the application through the non-overplayed allocation of memory objects onto the scratchpad memory while considering their conflict relationships in the cache memory.

IV. EXPERIMENTAL RESULTS The section evaluates the effectiveness of code repositioning, SPM code selection, Cache code selection and combined benefit of both along with external cache. The first section describes the simulation setting, while the next compare the power reduction and memory access distribution of benchmark programs Thus, this section compares code layout schemes determined using different methods

A. EXPERIMENTAL SETUP 1. Code repositioning for cache-only arrangement. 2. Code repositioning for SPM only pattern. 3. Code repositioning and SPM code selection for single and multiple processors. Code object is identified and linking is done using pre compiler and high level language. Code repositioning for cache only configuration works as follow 1) fetches the instruction from application 2) Searching for the data is done in cache, some energy is consumed for accessing the cache memory. The data will be searched in all the location of the cache memory and thus the energy is consumed for searching all the locations. If the data present in the cache then the process will start, else the process will go to the main memory to get the data. For accessing the main memory the energy consumption will increased because the processor accessed both the cache and main memory. Then the steps will be repeated till all the instructions complete. Following formula is used for calculating time and energy consumed.

cachet=(systemt-cachestartc) ---------------(1)

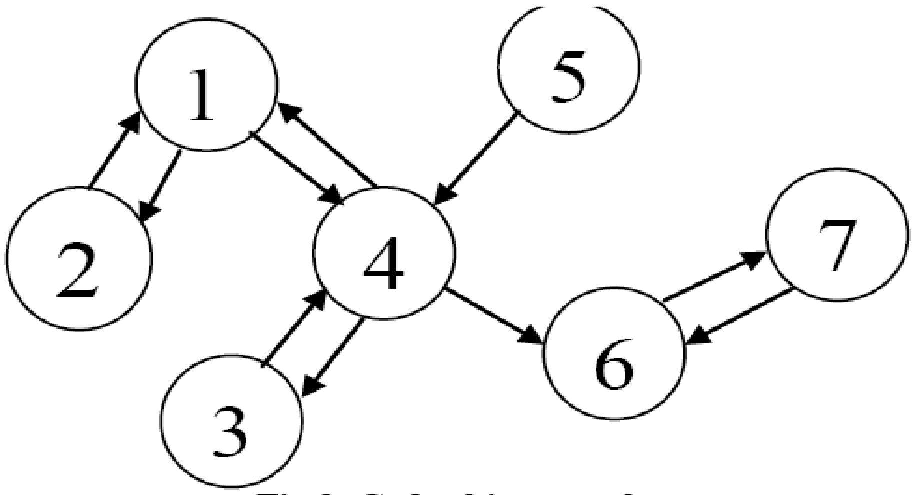
energycache=cachet*cachesize*vdd ---------------- (2)

Code repositioning for SPM only configuration works as follow fetch the instruction from application, search the data in SPM. Some energy is consumed for accessing the SPM. If the processor access the SPM, the processor can get the data directly by going to location without searching. So at first time itself processor can knew whether the data present or not. If the data is not present in the SPM then the process will go to the main memory, for accessing the main memory the energy consumption will increased because the processor accessed both the SPM and main memory. Then the steps will be repeat till all the instructions complete. Following formula

spmt=(systemt-spmstartc) ---------------------(3)

energyspm=(spmt*blksize*vdd) --------------------(4)

Code repositioning for cache and SPM configuration initially fetches the instruction for application. The searching of the data is done in cache, some energy is consumed for accessing the cache memory. If the data is present in the cache then the process will start, else the process will go for the SPM to get the data. So the energy will be consumed for accessing the SPM. Thus without accessing the Main memory the instruction is processed. This leads to the less consumption of energy, because for accessing the main memory only the energy will consumed more. Same is done for single and multiple processor alone with dynamic and static SPM and comparison graph is obtained. The searching of data is done using tabu search. Tabu search works in same way as traveling salesman problem that is finding the minimum path from the current memory location .Memory location of the code is similar to the graph shown in figure3.Minimum distance from the current location is identified and the code is brought through the shortest path. However, it is clear that having any code objects with size smaller than the SPM capacity is advantageous both in terms of less access energy and reducing conflicts with other code objects. Thus, this study proposes a metaheuristic such as Tabu search to avoid these traps.



$$\begin{tikzcd}
 & 1 \arrow[dl, shift left] \arrow[dr, shift left] & & 5 \arrow[dl] & \\
2 \arrow[ur, shift left] & & 4 \arrow[ul, shift left] \arrow[dl, shift left] \arrow[r] & 6 \arrow[r, shift left] & 7 \arrow[l, shift left] \\
 & 3 \arrow[ur, shift left] & & &
\end{tikzcd}$$

Fig 3: Code object graph

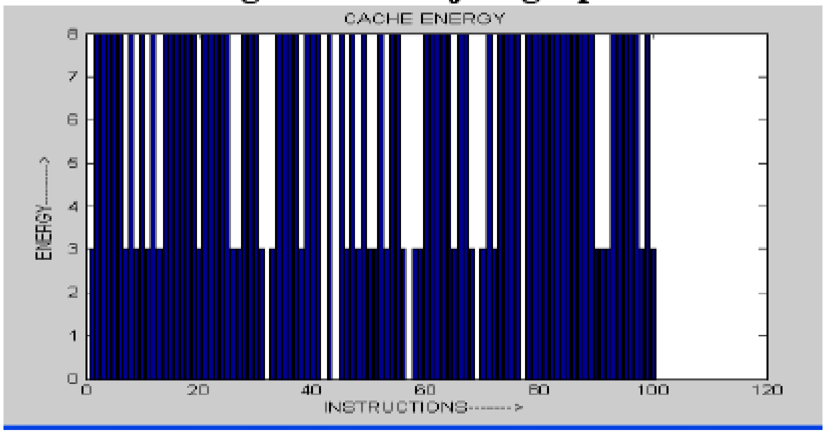


Fig 4: Energy consumption for cache only configuration

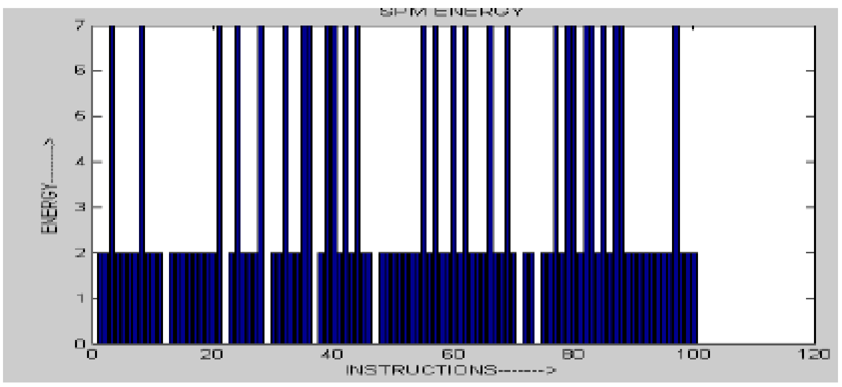


Fig 5: Energy consumption for SPM only configuration

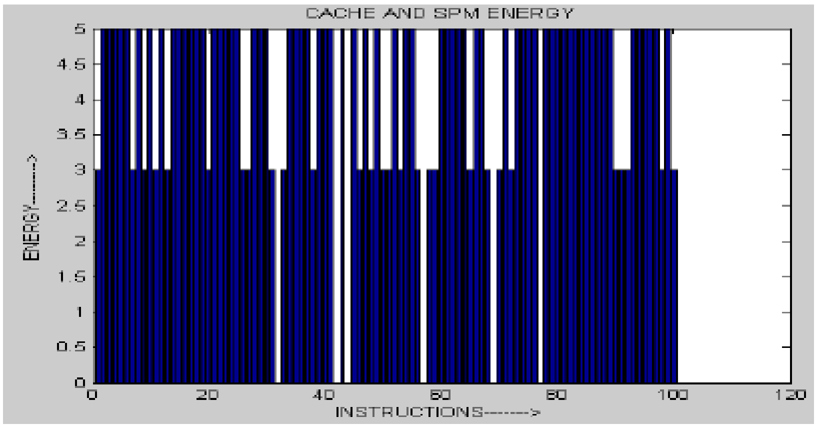


Fig 6:Energy consumption for SPM and cache configuration

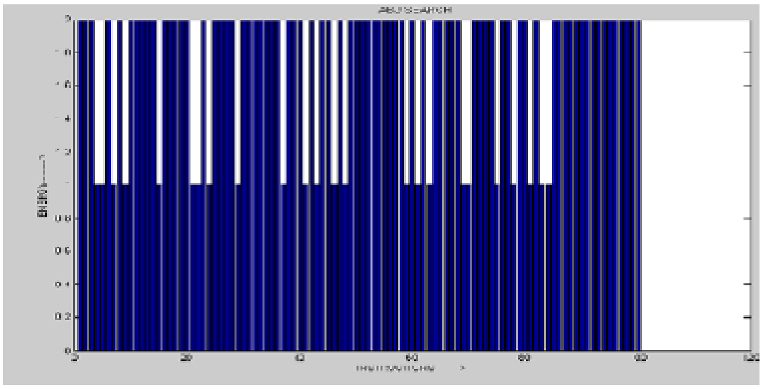


Fig 7:Energy consumption using Tabu Search

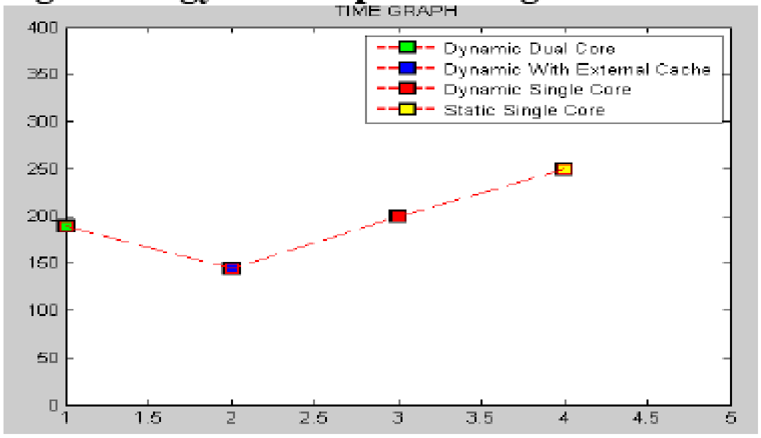


Fig 8:Energy *comparation* graph

V.CONCLUSION This paper presents an optimal code layout for embedded systems with a cache and SPM. The proposed models address layout problems under different memory configurations that for changing SPM and cache byte. Consequences show that layout firm using code repositioning and SPM code selection simultaneously offers better results than that using only the SPM code choice technique. The benefit of energy saving is manifested by either capturing the conflicting sets that occur relatively frequently in the SPM or rearranging the conflicting objects into less contend cache sets. The proposed method reduces power usage by reducing expensive cache misses, which also helps in improving performance of the system by about 3%. This method can be extended to get better result

REFERENCES:

[1]Stuti Ramola, “Digital Communication Technology and Advancements”, Advance in Electronic and Electric Engineering, Research India Publications, Vol. 2, 2014.

[2] Aleksandar Rodica, Miloš Jovanovica, Ilija Stevanovica, Branko Karanb, Veljko Potkonjakc,”Building Technology Platform Aimed to Development Service Robot with Embedded Personality and Enhanced Communication with Social Environment”, Digital Communications and Networks, 2015.

[3] Cristian S. Calude , “Modern Data Communications: Analog and Digital Signals, Compression, Data Integrity”, Modern Data Communication, August 2012.
[4] Dr. Allison F. Alden, Anthony G. Naglieri, “Reaching Real Time Moving Targets: The Use of Digital Communications to Inform and Mobilize College Students”, CSPA-NYS Journal of Student Affairs Vol. 12, Issue 1, 2012.
[5] Fernando Laguarda, “The Future of Digital Communications: Technical Perspective”, External Affairs and Policy Counselor Time Warner Cable.
[6] Vineet Kaul, “The Digital Communications Revolution“, Online Journal of Communication and Media Technologies, Vol. 2, Issue 3, July 2012.
[7] Jin Zhao, Xiaofeng Wu, “Application Of Digital Communication Techniques To Plastic Extrusion Process”, The Ninth International Conference on Electronic Measurement & Instruments ICEMI, 2009. [8] Asvin Gohil, Hardik Modi, Shobhit K Patel, “5G Technology of Mobile Communication: A Survey”, International Conference on Intelligent Systems and Signal Processing (ISSP), 2013.
[9] Gary Michel, Greg Pleinka,”Digital Communications For Relay Protection”, Working Group of IEEE Power System Relaying Committee.