Optimize Power Depletion to an Improvement Technique Of Embedded System

[1] R ALEKYA, [2] A SOUMYA
[1,2] Assistant professor, Department of Electronics and Communication Engineering,
St. Martin's Engineering College, Hyderabad-500100

Abstract- Embedded system with foreordained applications can accomplish further up degree by the expansion of quick and vitality effective scratchpad memory (SPM) on chip and moving regular gets to code or potentially information from primary memory to SPM. Numerous microchips invest larger part of its energy trusting that the information will show up from moderate memory gadgets associated with it. This is named as memory divider issue. The fundamental point of this paper is to lessen memory divider issue by speeding up the framework by processor speed as well as by the memory speed. Likewise this paper proposes a two-phase strategy which takes a shot at improving the productivity of the memory by concentrating on both on chip and off chip memory, along these lines diminishing the memory divider issue. Test results uncover 1) dispensing a devoted part of the on-chip processor memory as SPM 2) contemplating the consolidated advantage of store and SPM for both single and different processor 3) Using outer reserve alone with on chip SPM and reserve which can additionally limit the vitality utilization of the installed frameworks. Reproduction result for our technique lessens the vitality utilization of the framework by 3%.Energy increase can be acquired by utilizing an outside reserve alone with on chip store and SPM.

Keywords: Memory Wall Problem Embedded System, Cache, Scratch Pad Memory.

I. INTRODUCTION

Inserted framework is a convenient gadget .The primary structure thought of this kind of gadget is light weight and decreased force utilization. Two sorts of processor design are CISC (Complex Instruction Set Computers) and RISC (Reduced Instruction Set Computers).CISC utilizes greater program consequently consumes substantially more space and thus expend more capital than RISC. Installed framework which primarily centres on low vitality and light weighted item doesn't lean toward CISC for their turn of events. Subsequently we go for RISC which executes numerous orders in single activity. Various activities in single guidance lead to quick memory get to which thusly prompt the need of quick memory gadget as get and translate will happen quicker in RISC. So the use of RISC as a processor on inserted framework is exceptionally liked.

As of recently store is viewed as the memory which is for the most part used to give the straightforwardness between the processor and the memory. Truth is told reserves are as yet utilized in many universally useful processors. Numerous investigations have as of late proposed Scratch Pad Memory (SPM) which is a product controlled memory .SPM comprises of just information cluster and memory decoder rationale whereas store comprises of information exhibit, memory decoder rationale alongside label memory which makes it both bigger in size and force utilization than SPM, consequently SPM is effective than cache.SPM need an unequivocal help, for example, solid programming or some compiler coordinated device for their legitimate use , a solid programming rationale can utilize this memory. SPM devour 40% less vitality and 34% less space than reserve [1]. Execution time can be precisely anticipated in SPM. Developer can figure out what part of code or information ought to be set into the SPM, it utilizes uncommon fast memory circuit to hold little thing of information for

quick recovery. SPM streamlines storing rationale and assists framework with working without fundamental memory conflict particularly in reserve of MPSOC (multiprocessor framework 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 papers 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 a 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 0-1Knapsack 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 words 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.

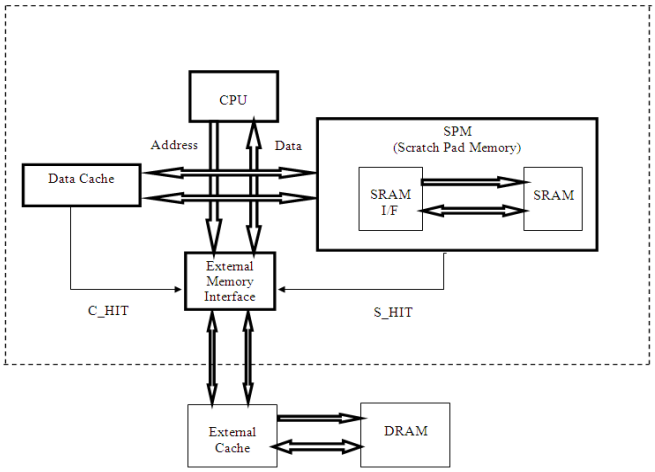

Fig1: Architecture Diagram

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.

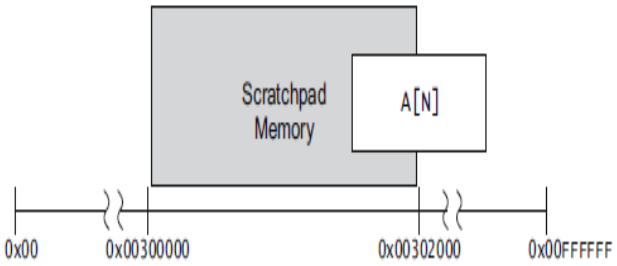

Fig 2: Processor Address Space Containing a Scratchpad Memory

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 employee. 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.

$$cache_t=(system_t\text{-}cachestartc) \quad (1)$$

$$energy_{cache}=cache_t*cache_{size}*vdd \quad (2)$$

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 accesses 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 is used for calculating time and energy consumption

$$spm_t=(system_t\text{-}spmstartc) \quad (3)$$

$$energy_{spm}=(spm_t*blk_{size}*vdd) \quad (4)$$

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 processors 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.

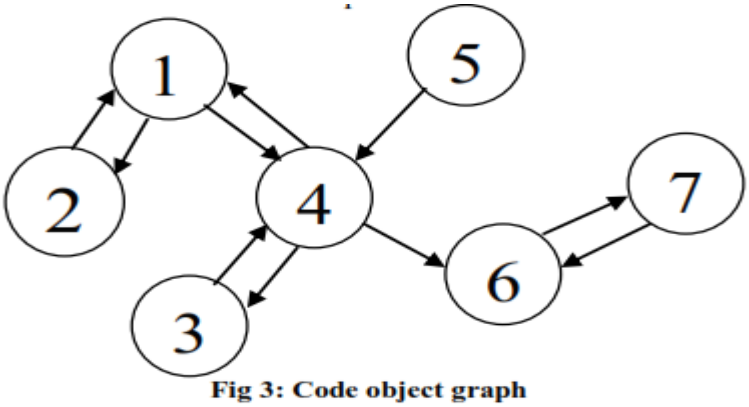

Fig 3: Code object graph

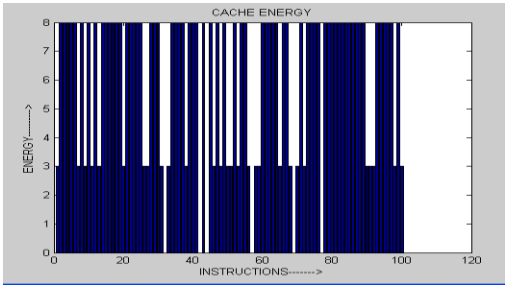

Fig 4: Energy consumption for cache only configuration

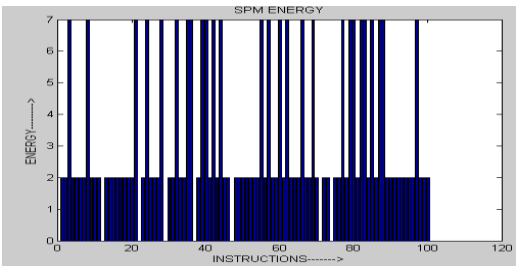

Fig 5: Energy consumption for SPM only configuration

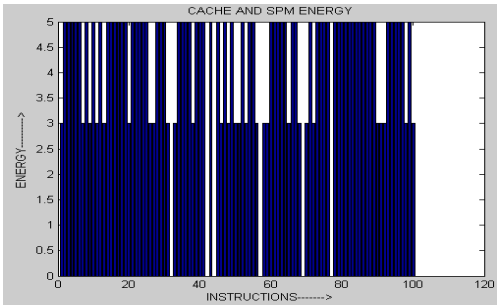

Fig 6:Energy consumption for SPM and cache configuration

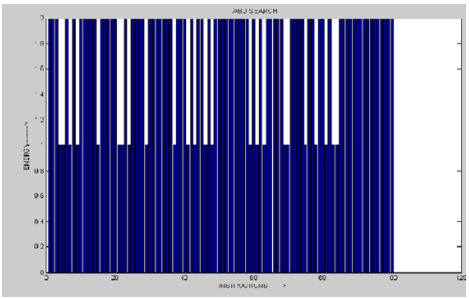

Fig 7:Energy consumption using Tabu Search

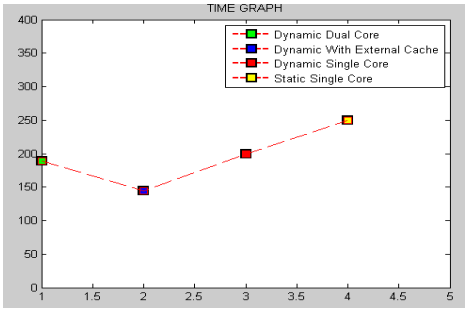

Fig 8: Energy *comparation* graph

B. BENCHMARK COMPARATION

SRAM budget splits between cache and SPM were tested. To assess the sensitivity of different methods toward energy and time consumption. Energy consumption levels were normalized to those of the corresponding original trace for each benchmark in which the entire on-chip SRAM was used. Plot obtained are normalized energy and time consumption for changing SPM and Cache byte.

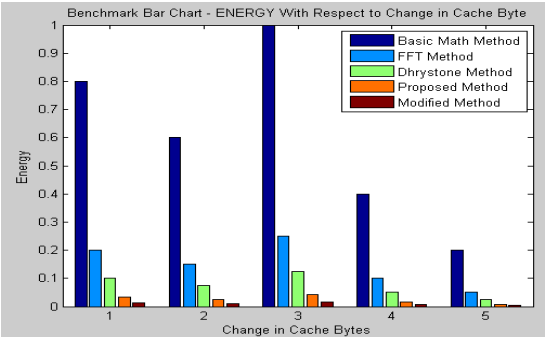

Fig 9: Normalize power usage with respect to change in cache byte

The normalized energy and time graph for various cache-SPM configurations appears as bars. The bench mark used is Basic Math method, FFT Method, Dhrystone Method which are compared with proposed and modified Method. The plot shows that power and time consumption of proposed method is less than that of existing system as well as the benchmarks in which entire SRAM is used.

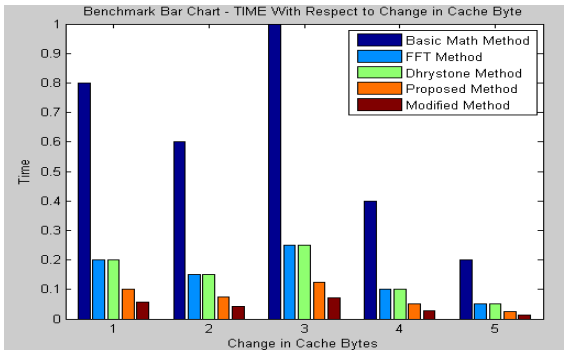

Fig 10:Normalized Time usage with respect to change in cache byte

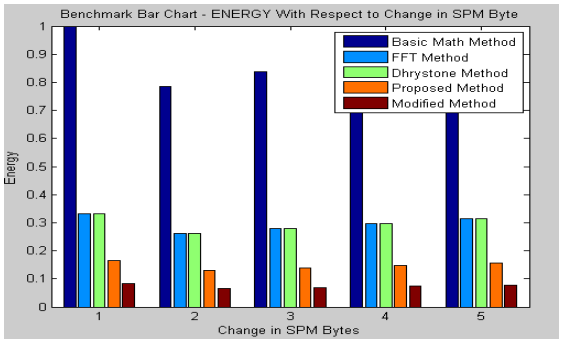

Fig 11:Normalized power usage with respect to change in SPM byte

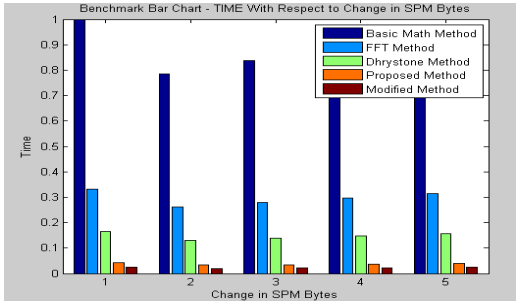

Fig 12: Normalized time usage with respect to change in SPM byte

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] Advanced Memory Optimization Techniques for Low-Power Embedded Processors by Manish Verma .Altera European Technology Center, High Wycombe, UK and Peter Marwedel University of Dortmund, Germany.

[2] ARM,http://www.arm.com/products/processors/classic/arm11/index.php, 2011.

[3] IBM, http://www.research.ibm.com/cell/, 2011.

[4] Freescale,http://www.freescale.com/webapp/sps/site/homepage.jsp.

[5] P.R. Panda, N.D. Dutt, and A. Nicolau, "On-Chip vs. Off-Chip Memory: The Data Partitioning Problem in Embedded Processor- Based Systems," ACM Trans. Design Automation of Electronic Systems, vol. 5, no. 3, pp. 682-704, 2000.

[6] Hyungmin Cho Bernhard Egger Jaejin Lee Heonshik Shin "Dynamic Data Scratchpad Memory Management for a Memory Subsystem with an MMU"pp 541-547.

[7] Y. Ishitobi, T. Ishihara, and H. Yasuura, "Code and Data Placement for Embedded Processors with Scratchpad and Cache Memories," J. Signal Processing Systems, vol. 60, pp.221-224, Aug.2010.

[8] S Wilton and Norm Jouppi: Cacti: An enhanced access and cycle time model, IEEE Joural of Solid State Circuit, May1996.

[9] R. Banakar, S. Steinke, B.S. Lee, M. Balakrishnan, and P.Marwedel, "Scratchpad Memory: A Design Alternative for Cache On-Chip Memory in Embedded Systems," Proc. 10th Int'l Symp. Hardware/Software Codesign (CODES '02), pp.73-78, 2002.

[10] J.L. Hennessy and D.A. Patterson, Computer Architecture: A Quantitative Approach, fourth ed. Morgan Kaufmann, 2006.

[11] M kandemir,associate member,IEEE,J.Ramanujam,Member IEEE,Mary Jane Irwin ,Fellow ,IEEE "A Complier Based Approach fro Dynamically Managing Scratch Pad Memories in Embedded System".

[12] M. Verma, L. Wehmeyer, and P. Marwedel, "Cache-Aware Scratchpad Allocation Algorithms for Energy-Constrained Embedded Systems," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 10, pp.2035-2051, Oct. 2006.

[13] Mahumut Kandemir,Associate Member ,IEEE,J Ramanujam,Member,IEEE,Mary Jane Irwin,Fellow,IEEE,N Vijaykrishna,Associate Member,IEEE,Ismail Kadayif, and Amisha Parikh" Compiler directed Scratch pad Memory Hierarchy Design and Management" .

[14] Tanja Van Achteren,Geert Deconinck,K.U.Leuven ESAT/ACCA,Belgium and F Catthoor ,Rudy Lauwereins "Data Reuse Exploration Technique for Loop dominated Application".

[15] Preethi ranjan panda and Nikil D Dutt " On chip Vs Off chip Memory:Data Partitioning Problem in Emb"edded Processor-Based Sysytems.

[16] D. Keitel-Schulz and N. Wehn, "Embedded DRAM Development:Technology, Physical Design, and Application Issues," IEEE Design and Test of Computers, vol. 18, no. 3, pp.
7-15, May 2001.