ISSN: 0974-5823 Vol. 7 No. 2 February, 2022

International Journal of Mechanical Engineering

Implementation of Efficient Load aware Memory Management (ELMM) Algorithm

K.Siva Sundari[#], R.Narmadha[*]

[#]*Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, India*

[1]Kalasiva2029@gmail.com

[*] *Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, India*

[2]narmadha.enc@sathyabama.ac.in

Abstract

Embedded system is considered as a variety of computer system has developed for certain dedicated application usually along real-time computing restraints. The recent and complex applications include multimedia streaming and signal processing makes dynamic memory allocation mandatory system. During the advancement of an embedded system, memory management becomes a major problem. The main difficult associated with memory allocation algorithm is decrease fragmentation, ELMM is proposed. First, the optimal clustering algorithm is utilized to separate and allocate the transient objects to heap memory which decrease memory fragmentation. Second, optimal signal-to-memory mapping algorithm is employed for hierarchical memory organization using evaluation of bounding windows. The proposed system computes the extra storage space using task monitoring algorithm. The results reveal that the proposed ELMM system could be obtained at a lower latency with minimal resource overhead and lower energy consumption. In the future, the presented technique can be developed for improving the efficiency which is utilized to improve the performance.

Key Words: ELMM, Optimal Clustering Algorithm, Optimal Signal-To-Memory Mapping Algorithm, Embedded system, FPGA memory.

1. Introduction

The Embedded System configuration has developed into a noteworthy field of current PC application, and the heading of now a day PC improvement. Implanted framework can use in numerous areas, for example, computerization field, car field, Portable gear, aviation, weapon types of gear just as different viewpoints throughout everyday life, so the Embedded framework has a decent prospect of application[1][2]. So as to adjust expanding decent variety and intricacy of the application, utilizing implanted working frameworks in the inserted frameworks has turned into a heading for the upcoming development of installed frameworks. Since utilizing the Real-time Operating System(RTOS) can be all the more soundly and proficiently to convey the CPU asset and different assets, rearrange the structure of utilization programming, abbreviate the hour of framework advancement, guarantee the ongoing exhibition and unwavering the system [3][4].

II. Background

Memory management is part of fundamental piece of any working framework. The Memory executives are dispensed or de allocates the piece of memory to the procedure according to the need [5]. In Static memory the executive is allotted at appropriate time and no progressions should be possible at run time [8], while in Dynamic memory, executive is allotted at run time. During procedure execution, memory allocation/de allocation should be possible in Dynamic memory, while in static administration this is not possible. The benefit of dynamic memory the executives requires less memory with safe administration [10]. Dynamic memory the executives

can be categorized into two types: 1) Manual Memory Management 2) Automatic Memory Management. In manual memory, the board developer can allocate or de-distribute memory physically according to the need, so its appeal is simpler, however, a little bit of leeway can cause bugs [7]. The board unit designates the memory if the article is out of extension while in programmed memory [9][6].

III. Methodology

The dynamic energy use of this technology depends significantly on the capability of the storages and the amount of the peripheral memory contained in the storage technology, as explained by the SRAM memories. In order to reduce dynamic energy consumption, it is possible to divide the address space of a computer's memory into several instances, each containing a portion of the address space that is frequently used [12]. To route read/write requests to the appropriate instances of memory, an interconnect, such as a bus or bespoke fabric, must be used when memory is divided into many units. In some cases, this link will negate the benefits of using split memory, as it consumes on-chip space and energy in and of itself. If many small memory modules are used instead of a single large memory module, more storage space will be required, resulting in an increase in cost. In order to maximize energy savings, the logically accessible address space must also be reorganized so that segments most frequently accessed are mapped to the same physical memory [13]. The majority of memory addresses utilised by the application are spread evenly throughout its address space; often and unusually, the same memory instances are mapped to the same memory instances, negating the advantage of split memory architecture. The challenge of finding the most appropriate memory arrangement in this situation becomes even more difficult.

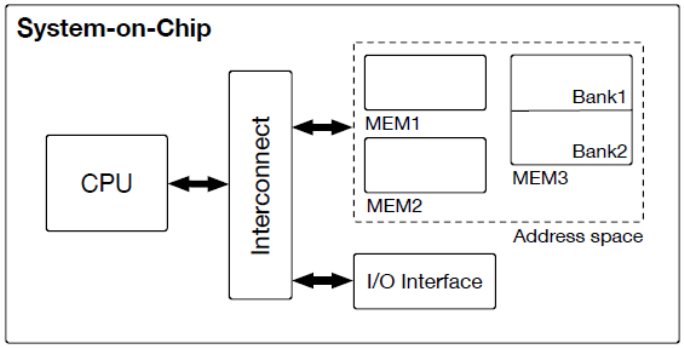

Fig.1 System architecture

In general, this optimization method is not suitable for multi-core SoC systems since it is designed for single-core systems with multiple on-chip memories, which is the primary focus of this optimization method. It is depicted in Figure 1 abstract system architecture for target platform, which is meant to be general in nature. In some cases, divided memories can also be referred to as scratchpad memories and are not divided [14]. In addition to the components that govern a system's communication with its environment, various other components of the system are mentioned as I/O interfaces, however, they do not contribute to the optimization process at all.

Static Power Optimization as an alternative for the optimisation model presented in this section which is aimed to reduce the dynamic energy use of a subsystem. SRAM low-power operation modes, as discussed in previous Section, provide significant optimization potential in this area. The application dependency graph (GD) is the most important input for determining a guided power mode activation schedule in guided power mode. The memory access frequency is collected and the locality of reference patterns between code and data profiles, which is unique to code and data profiles, is also encoded [15]. Optimal deployment, based upon the concept of static power decrease, maximises the time of the single memory blocks while limiting mode transitions. In accordance with the memory assignment Principle, this involves a memory subsystem with several memory blocks, each of which allows the operational mode in each memory block to be configured independently. As regards the binding method for the embedded software, application profiles coupled with any locality concept should, if possible, be able to map to the same storage instance [16]. This is crucial if a productive timetable is to succeed. Thus, highly linked code and data profiles should be stored in the same memory block and assigned the same or closely related C 2 C vector configuration memory.

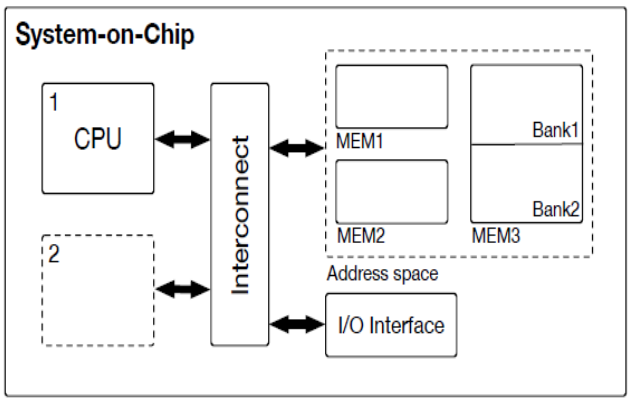

Fig. 2 Extended abstract target system architecture

Other than that, the number of operation mode activations and deactivations, combined with the time and energy consumed as a result, could more than outweigh the savings that would be realised by operating in low-power mode for a longer period of time. This optimization method is focused on a single-core SoC with on-chip memory, just like the previous one. Simplified system architecture is shown in Figure 2, which shows the optimization strategy in more detail. Both the static power optimization system model and the dynamic energy optimization system architecture described in Figure 2 contain two components which differ from the static energy optimization system architecture [17]. Both are necessary to take static power optimisation into account.

As a first stage, the degree of power consumption may be limited to a certain maximum value which can be justified by any co-processor. This is especially the case for designs with mixed signals which incorporate auxiliary components particularly sensitive to peak power.

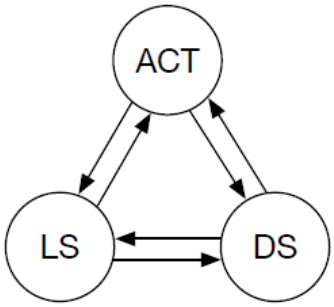

Fig. 3 Low-power mode transition graph

The second point is that support for low-power memory mode operations is included [18]. When a memory resource is employed in the system, it works for any memory in one of the operating modes, or in a transition between two of these conditions Figure 3 shows the states and transitions of the different operational modes of SRAM.

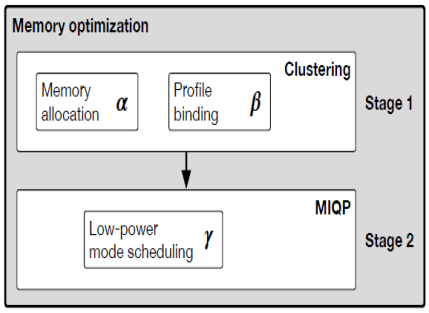

Fig. 4 Two-stage workflow for static energy minimization

Pictured in Figure 4 is the whole workflow for this two-stage implementation. It is intended to serve as a high-level summary representation of the procedure [19]. In addition to the solutions that have been proposed in the literature, it is feasible to estimate the amount of money that could be saved by reducing both dynamic and static energy use. Using a heuristic clustering method in stage 1 reduces the size of gigantic design spaces, followed by the formulation of a Mixed-Integer Quadratic Program MIQP in stage 2, allowing for handling of enormous design spaces.

Efficient Load aware Memory Management (ELMM)

It is a memory subsystem optimization proposal that is based on STT-RAM blocks. The proposed method is based on two properties of this memory technology that can be optimized. The needed logic to access SRAM memory results in a higher dynamic energy consumption [20]. The trade-off between STT-RAM write-operation and energy and latency is first and foremost. The task of identifying an energy-optimal collection of STT-RAM memories is characterised as an allocation and binding problem. As a result, the allocation of memory blocks of various dimensions is critical for energy usage during the reading and typing process, especially when combined with application segment binding and memory access speed considerations.

In addition to the operating frequency of the memory subsystem, it affects the overall system performance because the assignment of different operating voltage levels to different memory blocks in a single memory block influences the overall system performance [21]. It is feasible to trade higher system frequency for a less energy-efficient design based on the defined system boundaries. According to the 4 MiB STT-RAM memory operation mode in the 45 nm node, the maximum frequency of operations varies between 48 and 57 MHz, according to the penetration energy/latency compensation curve given in Figure 5.

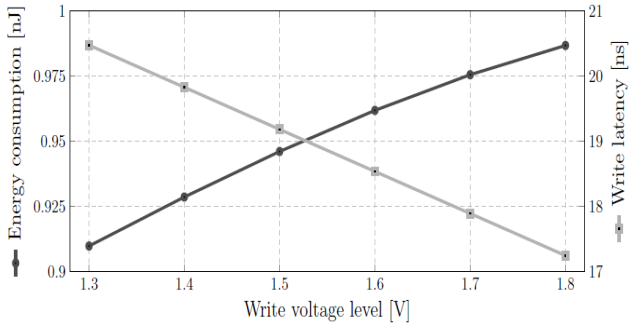

Fig. 5 Performance trade-off of a STT-RAM memory of 4 MiB on a 45 nm node

The write operation impacts different memory sizes offers even more design opportunities from frequencies of approximately 40MHz to more than 100MHz for smaller memories with only a few thousand bytes or less of store capacity [21]. The energy consumption varies simultaneously, which helps explain why it is not easy to develop a decent or even perfect solution in this field. Through the optimization approach described, different operating voltage levels and memory influence can be taken into account in STT-RAM memories, giving a realistic way to include these components at the same time.

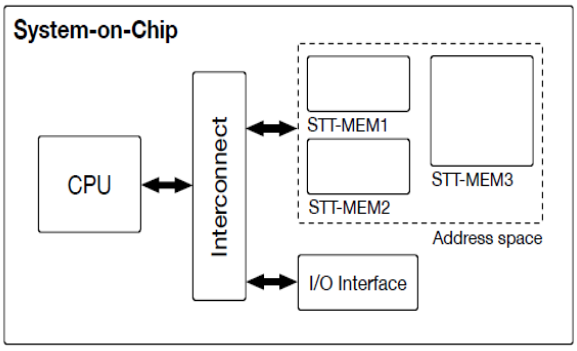

Fig.6 Abstract target system architecture

The use of shared memory variables and data structures is required due to a lack of accessible local storage space. This prevents the compiler from performing memory optimizations and allows the compiler to make the most efficient use of the processor registers available. This improves the efficiency with which CPU registers are utilised [22]. When accessing the shared memory pool, processor caches are bypassed in order to increase speed, which enhances performance overall. The data is promptly written back to shared memory after being requested by software, completing the transaction in the background. No matter the architecture of the processor, this algorithm can be executed.

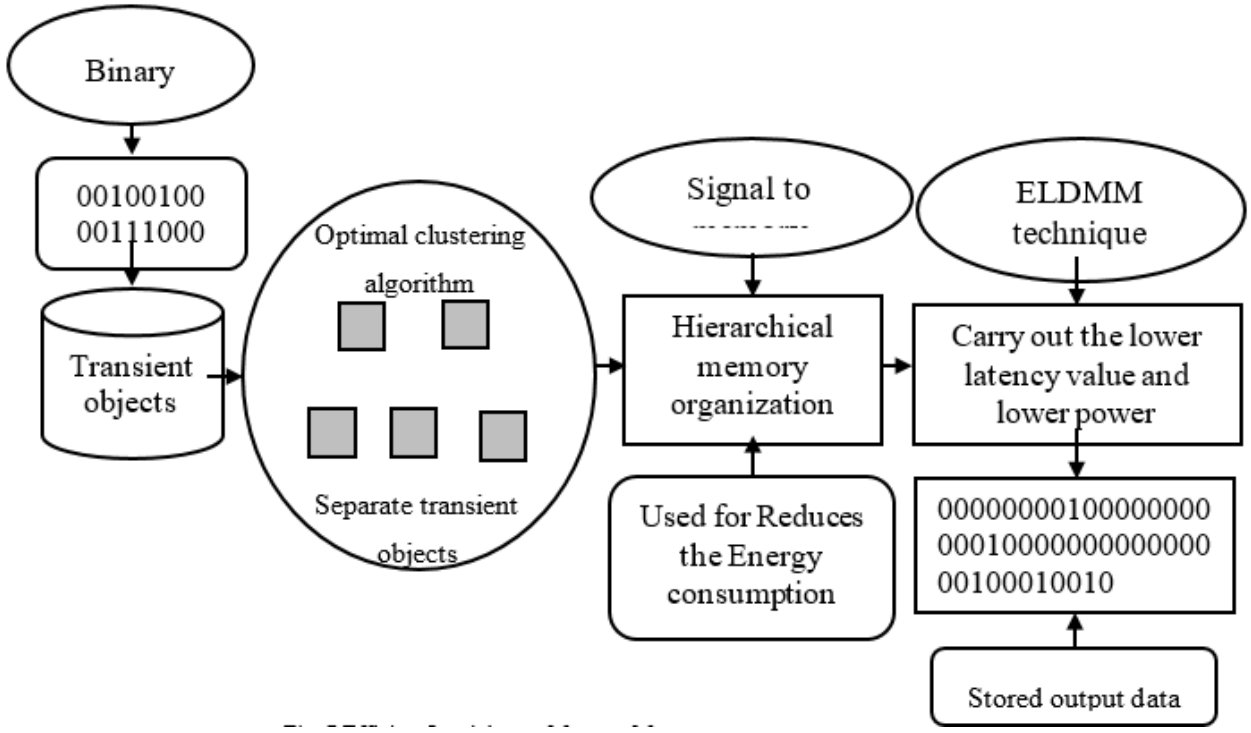

Fig. 7 Efficient Load aware Memory Management

Explicit instructions to flush all local storage into memory can be processed by the processor. During the development phase, it allows for unrestricted data flow between software and hardware. Shared memory, which guarantees data consistency and assures that it is preserved throughout the transaction by preventing data from being delayed in the processor's local memory [23].

As mapped memory access time is much greater than unmapped memory access time, using mapped memory instead of unmapped memory may result in a significant reduction in performance. To make these schemes as efficient and feasible as possible, factors such as processor bus speed, mapped memory speed, and mapped memory access logic speed must be considered [24].

IV. Results and Discussion

Using the Xilinx tool, the suggested ideal energy-efficient load-aware dynamic memory management system (ELMM) is implemented. Dynamic memory management is applied to and is simulated using the suggested optimal energy efficient load aware dynamic memory management system (ELMM). The basic memory of a computer is the heap, and a memory assignment capacity can dispense a touching square of memory of a specific size from it and return its position. To access that memory block, this location is stored in a pointer variable. The application retains the memory that is gradually released and restores it to the system.

User Environment	
OS	Microsoft Windows 7 , 64-bit
CPU Model	Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz
CPU Speed	3292 MHZ

Table 1 Memory Management User Environment Settings

Table 2 illustrates several important types of device usage statistics. The macro statistics declares about adder/ subtractor value, comparator value, multiplexer value, RAM and Register value. The miscellaneous presents a bond index value of aggregated term and number of buffer, simulation start-up time. The net statistics gives active sets, ground value, and power supply and pad input/output. Finally site usage represents input and output buffer value.

Device Usage Statistics			
Macro Statistics	Miscellaneous Statistics	Net Statistics	Site Usage
Adder/Subtractor =8	AGG_BONDED_IO=70	NumNets_Active=2031	BUFG=1
Comparator = 24	AGG_IO=70	NumNets_Gnd=1	CARRY4=42
Multiplexer = 152	NUM_BSFULL=460	NumNets_Vcc=1	FF_INIT=17
RAMs = 7	NUM_BUFG=1	NumNodesofType_Active_ PAD INPUT=33	IOB_INBUF=37
Register = 746	NUM_STARTUP=1	NumNodesofType_Active_ PAD OUTPUT=37	IOB_OUTBUF=33

Table.2 Device usage Statistics

The Figure 8 shows the simulated output for new memory size. In building the memory maps for each exhibit, our structure precisely relates to those pieces of clusters strongly got to, whose task to the on-chip layer yields the most notable benefit in terms of powerful vitality

Fig.8.Simulation results

Columns 1 of Table 3 display binary input values and binary output values, respectively. The memory sizes are handled as follows: the cross sections of the many clusters in the application are continually allocated to the ELMM in a certain order, increasing the memory size with discrete sums, while the typical number of gets to each exhibit component is decreased.

Binary Input	Binary Output
00100100	00000000100000000000100000000
00111000	00000000100010001000000000000
10000001	10000000000000000000000000001
00000000	00000000000000000000000000000
00001001	00000000000000001000000000001
01000000	00001000000000000000000000000
01100011	00001000100000000000000010001
00000000	00000000000000000000000000000
00001101	00000000000000001000100000001
01000101	00001000000000000000100000001

Table.3 Proposed EL MM Binary Input and Output values

V. Conclusion

This work proposes the ELMM system, which is the best energy-efficient load-aware dynamic memory management. The separation of temporary objects into heap memory and the reduction of memory fragmentation are both achieved through the use of an ideal clustering method. Also suggested is a signal-to-memory mapping technique that is best for hierarchical memory structure and computes bounding windows. The energy usage is decreased by computing the additional storage after the mapping process using the task monitoring technique. The experimental findings show that the suggested ELMM system outperforms existing techniques in terms of efficiency.

References

[1] Lopriore, L., 2016. Memory protection in embedded systems. Journal of Systems Architecture, 63, pp.61-69.

[2] Wang, Y., Xu, W., Yang, K. and Lin, J., 2012. Optimal energy-efficient power allocation for OFDM-based cognitive radio networks. IEEE Communications Letters, 16(9), pp.1420-1423.

[3] Hjertström, A., Nyström, D. and Sjödin, M., 2012. Data management for component-based embedded real-time systems: The database proxy approach. Journal of Systems and Software, 85(4), pp.821-834.

[4] Cheng, X.H., Gong, Y.M. and Wang, X.Z., 2009, March. Study of embedded operating system memory management. In 2009 First International Workshop on Education Technology and Computer Science (Vol. 3, pp. 962-965). IEEE.

[5] Ramanujam, J., Hong, J., Kandemir, M., Narayan, A. and Agarwal, A., 2005. Estimating and reducing the memory requirements of signal processing codes for embedded systems. IEEE transactions on signal processing, 54(1), pp.286-294.

[6] Łabiak, G. and Borowik, G., 2009. Statechart Diagrams Implementation in FPGA Structures with Embedded Memory Blocks. IFAC Proceedings Volumes, 42(21), pp.184-189.

[7] Wilson, P.R., Johnstone, M.S., Neely, M. and Boles, D., 1995, September. Dynamic storage allocation: A survey and critical review. In International Workshop on Memory Management (pp. 1-116). Springer, Berlin, Heidelberg.

[8] Wang, Y., Xu, W., Yang, K. and Lin, J., 2012. Optimal energy-efficient power allocation for OFDM-based cognitive radio networks. IEEE Communications Letters, 16(9), pp.1420-1423.

[9] Ye, L., Gniady, C. and Hartman, J.H., 2011, July. Energy-efficient memory management in virtual machine environments. In 2011 International Green Computing Conference and Workshops (pp. 1-8). IEEE.

[10] Masmano, M., Ripoll, I., Crespo, A. and Real, J., 2004, July. TLSF: A new dynamic memory allocator for real-time systems. In Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004. (pp. 79-88). IEEE.

[11] Goens, A., Castrillon, J., Odendahl, M. and Leupers, R., 2016. An optimal allocation of memory buffers for complex multicore platforms. Journal of Systems Architecture, 66, pp.69-83.

[12] Chang, D., Lin, I., Yong, L.: Rohom: Requirement-aware online hybrid on-chip memory management for multicore systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 36(3) (2017).

[13] J. Park, H. Seo and B. Kong, "Conditional-Boosting Flip-Flop for Near-Threshold Voltage Application", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 2, pp. 779-782, 2017.

[14] S. Lapshev and S. Hasan, "New Low Glitch and Low Power DET Flip-Flops Using Multiple C-Elements", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 10, pp. 1673-1681, 2016.

[15] Y. Li, H. Wang, R. Liu, L. Chen, I. Nofal, Q. Chen, A. He, G. Guo, S. Baeg, S. Wen, R. Wong, Q. Wu and M. Chen, "A 65 nm Temporally Hardened Flip-Flop Circuit", IEEE Transactions on Nuclear Science, vol. 63, no. 6, pp. 2934-2940, 2016.

[16] N. Kulkarni, J. Yang, J. Seo and S. Vrudhula, "Reducing Power, Leakage, and Area of Standard-Cell ASICs Using Threshold Logic Flip-Flops", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 9, pp. 2873-2886, 2016.

[17] F. Balasa, N. Abuaesh, C. Gingu, I. Luican and H. Zhu, "Energy-aware memory management for embedded multidimensional signal processing applications", EURASIP Journal on Embedded Systems, vol. 2017, no. 1, 2016.

[18] Se-Jun Kwon, Sang-Hoon Kim, Hyeong-Jun Kim &Jin-Soo Kim 2017, LZ4m: A Fast Compression Algorithm for In-Memory Data‟, in IEEE International Conference on Consumer Electronics (ICCE).

[19] YunchengGuo, Yu Hua 2018, „DFPC-Dynamic Frequent Patten Compression scheme in NVM based main memory‟, in Design, Automation & Test in Europe Conference, pp. 1622-1627.

[20] EshaChoukse&MattanErez 2018, „Compress points: An Evaluation Methodology for Compressed memory systems‟, IEEE Computer Architecture Letters, pp. 126-129

[21] David Kaeli 2017, Dual Dictionary compression for last level cache‟, IEEE international conference on computer design, pp. 353-360.

[22] Sparsh Mittal 2016, A survey of Architectural approaches for data compression in cache & main memory systems‟, in IEEE transactions on parallel & distributed systems, pp. 1524-1536.

[23] Enemali, G, Adetomi, A & Arslan, T 2017, 'FAReP: Fragmentation Aware Replacement Policy for Task Reuse on Reconfigurable FPGAs', IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 202-206.

[24] Kim, Y 2013, 'Power-efficient configuration cache structure for coarse-grained reconfigurable architecture' Journal of Circuits, Systems and Computers, vol. 22, no. 03, pp.1350001.