International Journal of Trend in Scientific Research and Development, Volume 1(2), ISSN: 2456-6470
www.ijtsrd.com

Effective Analysis on Power and Memory Optimization Techniques for Embedded Systems

Mini R. S.
Lecturer in Electronics, Department of Electronics Engineering,
Government Polytechnic College, Kottayam, Kerala, India

ABSTRACT
Memory systems have been identified as a major source of embedded software performance and power consumption. Because embedded systems require low power consumption, it is worthwhile to investigate memory optimization techniques for these systems. There are numerous techniques in the literature for reducing embedded system power/energy consumption and execution time. This paper revisits and discusses these techniques, with an emphasis on hardware and software optimizations. Reduced power dissipation in all parts of the design and at all stages of the design process is required for power-efficient design, subject to constraints on system performance and service quality. To meet these often-conflicting design requirements, power-aware high-level language compilers, dynamic power management policies, memory management schemes, bus encoding techniques, and hardware design tools are required.

Keywords*: Power, Memory Optimization Techniques, Embedded Systems, Power-aware software compilation, Dynamic power management*

Introduction
There are two types of computer systems: general purpose systems and special purpose systems. General-purpose systems can be used for a variety of purposes. The applications of general-purpose systems are not predetermined. Personal computers with Intel *86 architectures are a good example of general purpose systems.

These systems are expected to perform a variety of tasks with reasonable performance, which means that if the application can be completed in a reasonable amount of time, it will be considered acceptable. As technology advances, millions of circuits can be integrated on a single chip, allowing general purpose systems such as workstations and personal computers to play a significant role in computing environments.

However, general purpose systems cannot be used in some application domains due to both their performance and their cost. [1-2]

General purpose systems are not considered a competitive solution in some areas, such as telecommunications, multimedia, and consumer electronics. Special purpose systems have specific application domains whose requirements for real-time performance and compact size must be met at any cost, even if it means removing some system features [3]. For example, if special purpose systems in a cellular phone cannot meet real-time performance requirements, the output will be inaudible.

Structure of Embedded Systems
As shown in Figure 1, an embedded system is a typical design methodology for a special purpose system, consisting of three main components: an embedded processor, on-chip memory, and synthesised circuit. An embedded system's hardware and software are specially designed and optimised to solve a specific problem. From a manufacturing standpoint, implementing an entire system on a single chip, also known as system-on-a-chip architecture, is profitable. Embedded systems have strict size constraints because their cost is heavily dependent on size. Memory is the most important component in embedded system size. It is critical to minimise memory size by optimising its usage in order to reduce costs. In embedded systems, memory is divided into two parts: program-ROM and data-RAM.

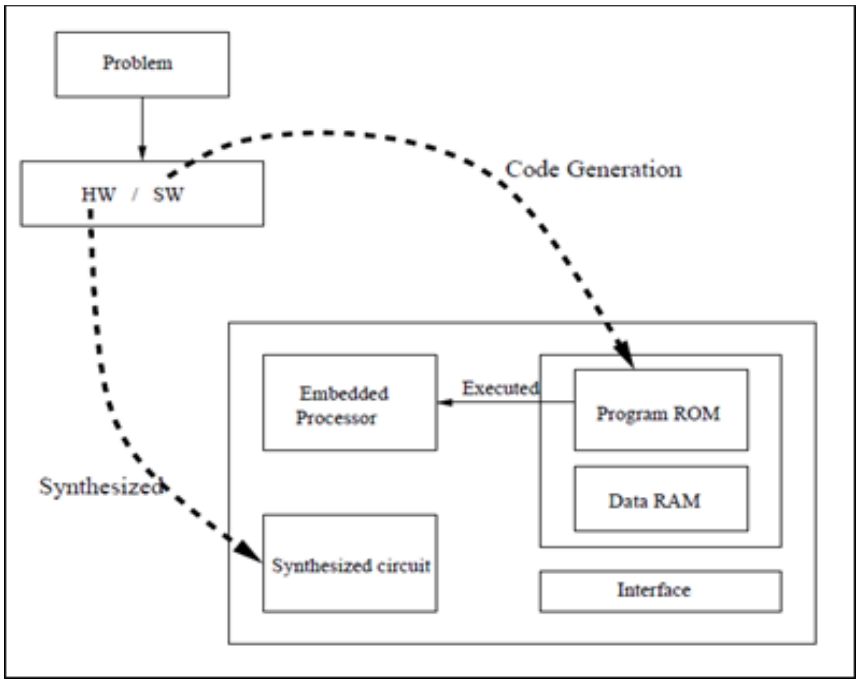

Figure 1: Structure of embedded systems

Advantages of Embedded Systems

The following are the benefits of embedded systems.

time-to-market For an embedded processor, there are numerous special purpose processors available. Only the time-critical components of an application are synthesised into a customised circuit, reducing the complexity of designing embedded systems. The use of high-level languages boosts the productivity of the software implementation phase. flexibility New standards emerge as technology advances. Video coding standards, for example, evolved from JPEG to MPEG1, MPEG2, and MPEG4. This application change will be accommodated by rewriting software rather than redesigning an entire embedded system [76, 63]. As a result, embedded systems are well suited to application evolution. This adaptability contributes to a short time-to-market cycle and low cost [5].

performance in real time Implementing time-critical tasks in a synthesised circuit aids in achieving high speed. If this objective cannot be met, the application should be reanalyzed and partitioned. The optimization technology used to generate high-quality (fast) code is critical to achieving this goal.

low price There are many low-cost special-purpose processors available when compared to general-purpose processors. The use of off-the-shelf special purpose processors and synthesising only time critical parts into hardware contributes to the low cost of embedded systems. It is critical to generate compact code in order to reduce costs by optimising on-chip memory usage.

Compiler Optimization for Embedded Systems

Special purpose processors with embedded capabilities differ from general purpose processors in several ways. DSPs, for example, have specific functional blocks that are optimised for common signal-processing algorithms. A typical example is a multiply accumulation (MAC). DSPs are distinguished by their irregular data paths and heterogeneous register files [6]. DSPs have limited data paths to save money and space. It is not uncommon for a specific register to be dedicated to a specific function block with this irregular data path topology, which means that the input and output of a function unit were fixed at the time the DSPs were designed. Figure 1.4 depicts the TMS320C25, a Texas Instruments DSP series. There are three registers that have specific uses. A multiplier, for example, requires one of its operands to be from the t register, and the result to be stored in the p register. The output of an ALU should be stored in an accumulator. As a result, each register should be treated differently. The data path is constrained.

Review of Literature:

Several works, including [7], [8], and [9], show that the Instruction Memory Organization (IMO) and Data Memory Hierarchy (DMH) account for a sizable portion of the chip area as well as the energy consumed by the system. According to [6,] both memory architectures now account for up to 40-60% of an embedded instruction-set processor platform's total energy budget.

Memory consumes more than half of the silicon real-estate in systems-on-chip (SoCs). These are products that combine multiple components into a single system on a single silicon chip.

Off-chip memory accesses dominate the power consumption of signal processing-oriented embedded systems, according to Catthoor et al. [10]. Even excessive cache behaviour can result in significant excess power consumption in some cases.

Objectives:

- Power and Memory Optimization Techniques for Embedded Systems
- Defining Embedded System Architecture
- Memory Optimization Techniques
- Embedded System Hardware and Software Techniques

Research Methodology:

Result and Discussion:

Ge et al. [11] depict a typical embedded system architecture that includes a processor core, reconfigurable hardware, instruction cache, data cache, on-chip scratch memory, on-chip DRAM, and off-chip memory in Figure 2. The computations are partitioned into different computational units, as shown in this figure, while the data is assigned to different storage components.

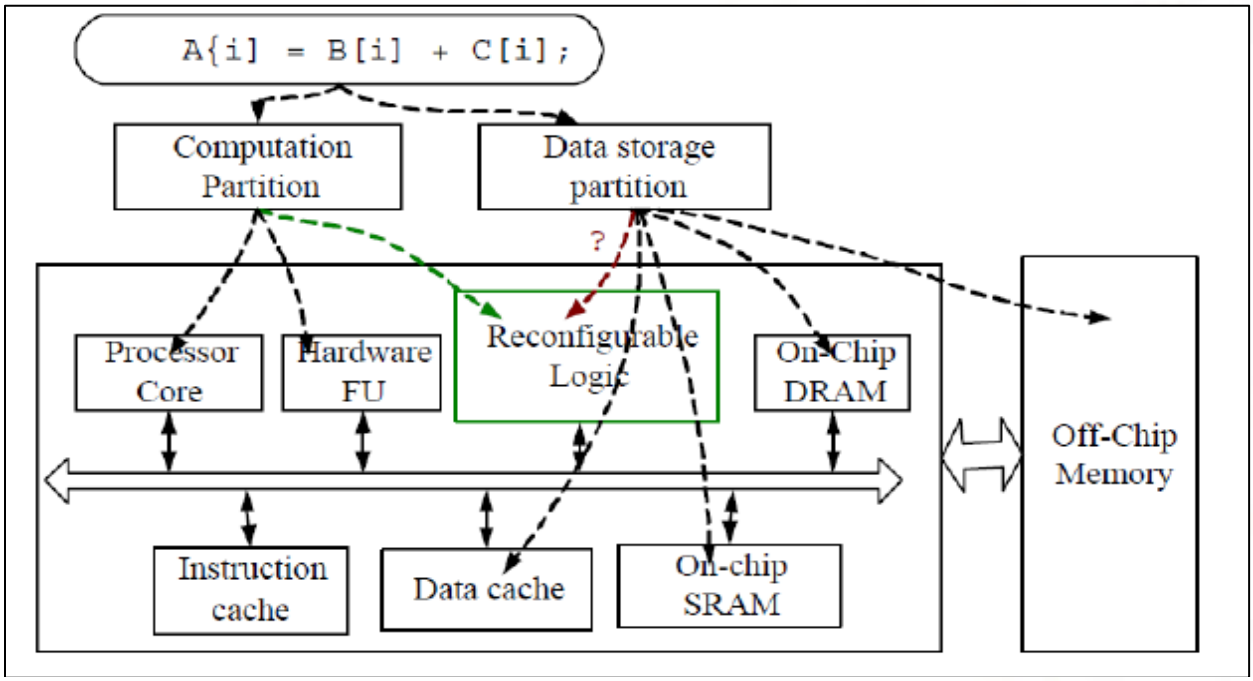

Figure 2 depicts a typical embedded system architecture [11].

Off-chip, on the other hand, may employ a variety of technologies, including synchronous dynamic random access memory (SDRAM) or Rambus DRAM. Caches, on-chip main memory, and off-chip main memory can all have an impact on embedded system performance and power/energy consumption.

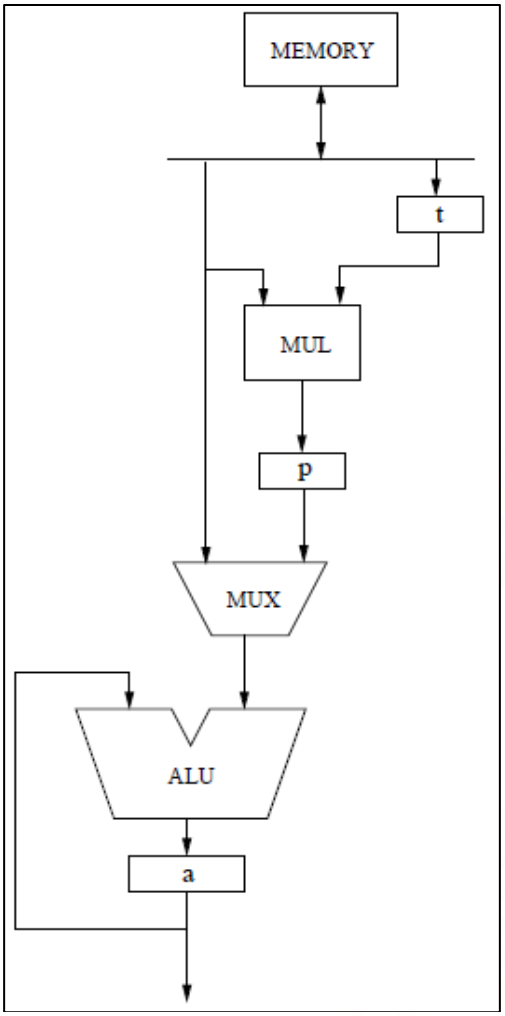

Figure 3: TI TMS320C25

MEMORY OPTIMIZATION TECHNIQUES

There have been numerous proposals for memory system optimizations. These types of memory optimizations can be classified into several categories. However, the traditional approaches are generally hardware and software optimizations, but both can be combined. This paper concentrated on traditional approaches. Figure 4 depicts a taxonomy of low-power memory design approaches found in the literature [12] [13].

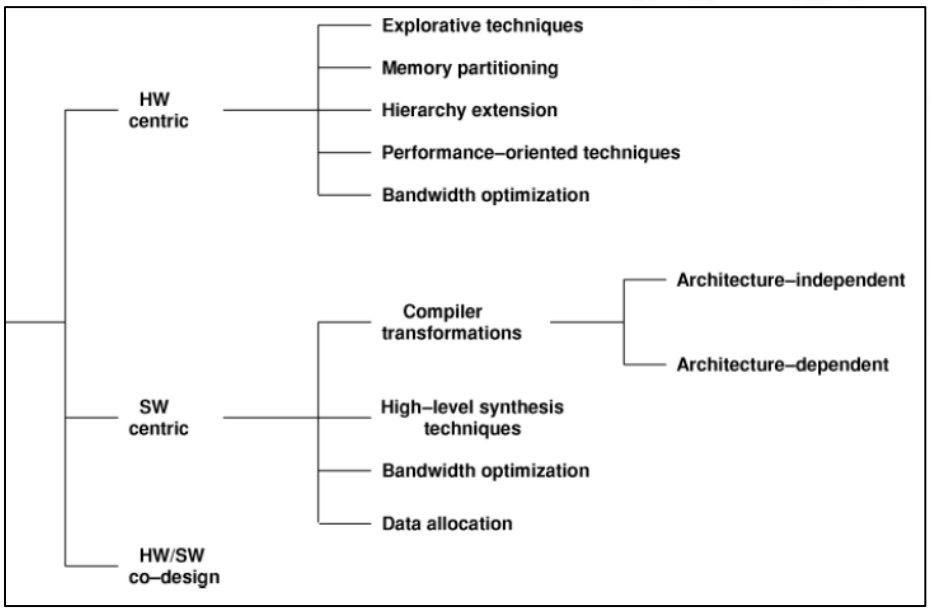

Figure 4: A classification of memory optimization techniques [12].

Benini et al. [12] describe techniques and methodologies for optimising the energy consumption of a hierarchical memory subsystem that have been presented in the literature. According to the authors, the goal of energy-efficient memory design is to reduce the overall energy cost of memory access while maintaining performance and size constraints. By allowing access to smaller banks, hierarchical organisations save memory energy. Furthermore, because most applications are characterised by nonuniform memory access patterns, mapping frequently accessed locations to low hierarchy levels aids in lowering total energy consumption.

Panda et al. [13] present data memory optimization techniques for embedded systems. Topics such as code transformations and memory addressing were thoroughly discussed using an original classification: platform-independent memory optimizations that operated on a source-to-source level, and platform-dependent optimizations that could be applied to memory structures at various architectural granularities. The disadvantage of this paper is that it does not address the context of parallel platforms, such as task-level parallelism.

Hardware and software in embedded systems can be custom-designed and optimised for specific applications. To optimise energy for an application, we must first solve a number of problems. What are the best PCM and DRAM sizes, for example, how to statically allocate data and programme, and how to migrate data between PCM and DRAM? To fully explore this application-specific PCM/DRAM main memory for energy optimization in embedded systems, many new optimization problems arise, and various system-level optimization techniques must be developed.

Shao et al. [14] present a hybrid memory system architecture in which PCM is used to replace DRAM as much as possible in order to reduce system energy by utilising PCM's lower standby power. Figure 5 depicts an embedded system system architecture with hybrid PCM/DRAM main memory. NAND flash is used as a secondary storage medium for storing user data. The main memory is a hybrid PCM/DRAM memory. Unlike DRAM-based main memory, this hybrid main memory can store code as well as user data due to the non-volatility of PCM.

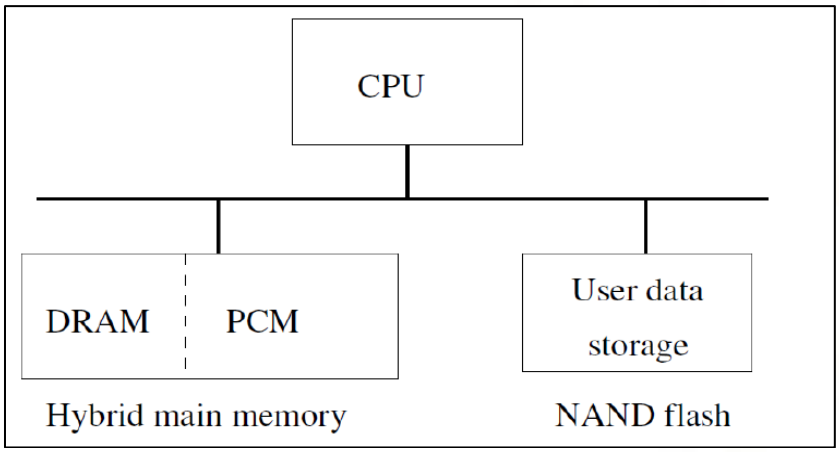

Figure 5: System architecture in embedded systems with application-specific hybrid PCM/DRAM main memory [14].

Power-aware software compilation

Based on instruction-level characterization and simulation of the underlying hardware, power analysis techniques for embedded software have been proposed. Statistical profiling techniques have been proposed in to improve the efficiency of software power analysis. The SOC design paradigm has fueled research into hardware and software power consumption co-estimation. It has been reported that embedded software power consumption can be reduced through compiler optimizations, source-level transformations, and memory management schemes. [15-16]

Dynamic power management

Dynamic power management, which refers to the selective shutoff or slowing down of idle or underutilised system components, has proven to be a particularly effective technique for reducing power dissipation in such systems. Incorporating a dynamic power management scheme into an already complex system is a difficult

process that may necessitate many design iterations as well as careful debugging and validation. A dynamic power management policy's goal is to reduce an electronic system's power consumption by putting system components into different states, each representing a different level of performance and power consumption. Based on the system's history, workload, and performance constraints, the policy determines the type and timing of these transitions.

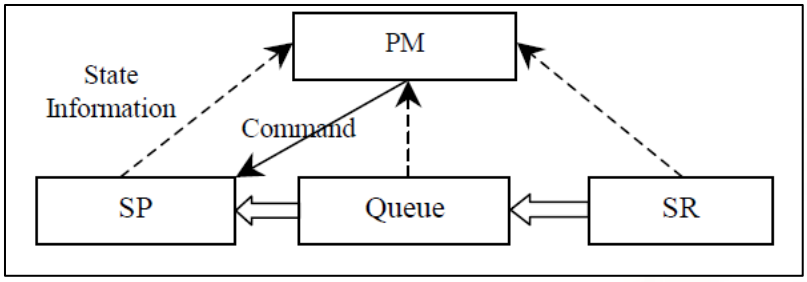

Figure 6: A model of a power-managed system.

A power manager (PM), a service provider (SP), a service requestor (SR), and a service request queue (SQ) comprise the system model (cf. Figure 6). [17]

Conclusion:

This paper reviewed memory optimization techniques for embedded systems. First, we provided some background on embedded systems, their characteristics, and some notes on memories. Following that, we reviewed some memory optimization techniques and presented a taxonomy of low-power memory design approaches found in the literature. This paper also reviewed techniques and tools for power-efficient embedded system design, taking into account the hardware platform, application software, and system software.

References:

[1] P.R. Panda. Memory Optimizations and Exploration for Embedded Systems. PhD thesis,

[2] UC Irvine Dept. of Information and Computer Science, 1998.

[3] J. L. Hennessy and D. A. Patterson. Computer Architectures: A Quantitative Approach.

[4] Morgan Kaufmann, 1996.

[5] J. G. Ganssle. The Art of Programming Embedded Systems. Academic Press, Inc., San Diego, California, 1992.

[6] G. Goossens, F. Catthoor, D. Lanneer, and H. De Man. Integration of Signal Processing Systems on Heterogeneous IC Architectures. In Proceedings of the 6th International Workshop on High-Level Synthesis, pages 16-26, November 1992.

[7] S. Devadas, A. Ghosh, and K. Keutzer. Logic Synthesis. McGraw Hill, New York, NY, 1994.

[8] E. A. Lee. Programmable DSP Architectures: Part I. IEEE ASSP Magazine, pages 4-19, October 1988.

[9] F. Catthoor, P. Raghavan, A. Lambrechts and M. Jayapala, Ultra-low energy domain-specif instruction-set processors, Berlim: Springer, 2010.

[10] J. L. Hennessy and D. A. Patterson, Computer Architecture – A quantative approach, 4ª ed., San Francisco: Morgan Kaufmann, 2007.

[11] M. Verma and P. Marwedel, Advanced memory optimization techniques for low-power embedded processors, Berlim: Springer, 2007.

[12] F. Catthoor, E. De Greef, S. Wuytack, L. Nachtergaele, F. Balasa and A. Vandecapelle, Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design, Boston: Kluwer, 1998.

[13] Z. Ge, H. B. Lim and W. F. Wong, "Memory Hierarchy Hardware-Software Co-design," Journal of Computer Science, pp. 1-9, 2004.

[14] L. Benini, A. Macii and M. Poncino, "Energy-aware design of embedded memories: A survey of technologies, architectures, and optimization techniques," Journal of ACM Transactions of Embedded Computing System, vol. II, no. 1, pp. 5-32, 2003.

[15] P. R. Panda, F. Cathoor, N. D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandercappelle and P. G. Kjeldsberg, "Data and

memory optimization techniques for embedded systems," pp. 149-206, April 2001.

[16] Z. Shao, Y. Liu, Y. Chen and T. Li, "Utilizing PCM for Energy Optimization in Embedded Systems," in ISVLSI, Amherst, MA, 2012.

[17] embedded software: A first step towards software power minimization," Proc. Int. Conf. Computer-Aided Design, Nov. 1994.

[18] T. Sato, Y. Ootaguro, M. Nagamatsu, and H.Tago, "Evaluation of architecture-level power estimation for CMOS RISC processors," Proc. Int. Symp. Low Power Electronics, pages 44-45, Oct. 1995.

[19] C.-T. Hsieh, M. Pedram, G. Mehta, and F.Rastgar, "Profile-driven program synthesis for evaluation of system power dissipation," Proc. Design Automation Conf., pages 576-581, June 1997.