Asian Journal of Applied Science and Technology (AJAST)
Volume 8, Issue 3, Pages 32-42, July-September 2024

Exploring Embedded System Tackling Design Challenges Methodology and Optimization Strategies

Mrs. S. Saranya[1*], Mr. M. Arul Pugazhendhi[2] & Dr. S. Mohamed Nizar[3]

[1]*ME-Applied Electronics, IFET College of Engineering, Villupuram, Tamilnadu, India.*
[2,3]*Department of Electronics and Communication Engineering, IFET College of Engineering, Villupuram, Tamilnadu, India.*
Corresponding Author (Mrs. S. Saranya) Email: saranyashiva1207@gmail.com[*]

DOI: https://doi.org/10.38177/ajast.2024.8304

Article Received: 11 May 2024 *Article Accepted: 23 July 2024* *Article Published: 28 July 2024*

ABSTRACT

The current design methodologies are no longer relevant due to the complex nature of embedded products and the need for quick development. The relevance of the present state of embedded system technology is highlighted, emphasizing the need to resolve design problems, implement optimization techniques, and create development processes for embedded systems. We fully focus on the safety and security of your design when we are solving a problem, especially for embedded systems applications that are critical to mission success. A strong understanding of embedded system hardware and software design is necessary in order to implement the strategies and methodologies of embedded system development. This is due to the fact that embedded systems comprise both software and hardware. These kinds of constraints include things like time constraints that are applied in real time, interactive designs, validation, and extensive testing.

Keywords: Embedded system; Microcontroller; Testing; ASIC; Embedded system development; Validation.

1. Introduction

Microcontroller and microprocessor-based systems are an absolute necessity, especially for real-time applications that operate on a tight budget. These specialized computer systems are furnished with sensors and other essential parts that are needed to accomplish particular tasks successfully. Design and prototyping iterations must be repeated in order to optimize embedded systems' effectiveness and efficiency. Frequent encounters with design issues often impede the development of embedded systems. The main focus areas of this investigation into embedded systems are design issues, methodology, and optimization techniques. The microcontroller or processes have to take into account a number of technological limitations, including memory management, power consumption, energy efficiency, and real-time applications. These are but a handful of the restrictions. Many obstacles in hardware and software design must be overcome in order to optimize and develop firmware or software for embedded systems in a dependable manner. When working with limited resources, code optimization is crucial to reducing its size and enhancing its performance. One way to achieve this is to remove any extraneous code.

In this study, the Y-chart method is introduced as a tool for developing programmable architectures. With the aid of these concepts, designers can carry out a methodical investigation of the architectural design field [1]. Our goal at the Gigascale Silicon Research Centre is to develop new approaches that prioritize re-use across all abstraction levels in order to address these issues.In order to attain at least some of the productivity gains mentioned above, we put forth a number of fundamental principles for system design [2]. We provide a memory exploration method that considers three performance metrics: cache size, processor cycle count, and energy consumption, to aid in this design decision. Performance is demonstrated to be impacted by cache variables such as line size, cache size, set associativity, tiling, and off-chip data organization [3]. Embedded processor designers have started integrating virtualization extensions into their processor architectures as cloud computing has grown in popularity. The integration of these improvements into embedded systems is proof of the long-ago convergence of cloud computing

and virtualization techniques [4]. The increasing proximity of fabrication technologies to the nanoscale presents computer architects with a plethora of new physical design challenges. Unpredictable production processes, diverse environmental factors, complex designs, and sporadic disruptions all compromise the accuracy and dependability of systems. We demonstrate the advantages of using typical-case optimization (TCO) techniques to increase the design flexibility of the essential components of the adder circuit.

Let's first discuss the background before diving into the benefits and drawbacks of using computer-aided design (CAD) tools for better-than-worst-case design [5]. Energy consumption has always been an important consideration in the design of some systems, like sensor networks and devices with a short battery life. The importance of this issue has grown over time in other kinds of systems, where performance enhancement has always been the main focus, like data centers and supercomputers. However, the widespread acclaim for performance improvements often overlooked the fact that energy consumption increased significantly [6]. The lifespan and energy output of batteries are significantly impacted by two important phenomena: the Rate Capacity effect and the Recovery effect. Both of these phenomena are well captured by our stochastic battery model. By inserting parameters related to the physical properties of the electrochemical cell, we can simulate how the battery operates by using mathematical models [7]. We present a platform-based design methodology that encompasses all stages of the cyber-physical system (CPS) design process by using contracts to precisely define and abstract its components. The application of contracts forms the foundation of this methodology. One characteristic of an iterative design process is that it starts with a high-level specification and ends with the creation of a low-level implementation that utilizes a number of pre-existing components [8]. SymTA/S analyzes the system's timing and performance by using formal scheduling analysis techniques in conjunction with symbolic simulation. The tool can handle a broad variety of architectures, complex task dependencies, and context-aware analysis, among other things [9]. Measuring the system's overall performance is the responsibility of the software. The least advantageous scheduling scenarios, bus and processor utilization, and start-to-finish latencies are all included in this category. Furthermore, Sym TA/S integrates optimization algorithms with system sensitivity analysis to facilitate quick design space exploration [10]. This makes the design process more effective.

Our group has created an operating system we call Tiny OS, a set of tiny RF wireless sensor devices, and a networking architecture for low-power, low-capacity devices that can operate in dynamic, self-organizing, and interactive environments. Since the study of microscopical computing is still in its infancy, we would like to learn more about the design procedures that are applied in these kinds of applications [11],[12]. Based on all available data, it appears that designing systems using three-dimensional technologies presents a wide range of opportunities. It is unquestionably the most promising chance to stop the semiconductor industry from adopting Moore's law, especially in light of the recent extraordinary breakthroughs that these three-dimensional technologies have made [13]. The goal is to find a cost-effective design implementation that meets all requirements, including goals, constraints, and functionality, in order to complete the partitioning task. Traditionally, the decision of which system blocks could be built as hardware and which could be run as software on a standard processor fell to the system designer, who depended on their experience in the field. This paper delves deeply into the investigation of different system partitioning strategies. The most crucial step in applying the different approaches successfully is to create a

model that is flexible enough to handle a range of co-design problems and to identify the best solution for each unique issue. There has been an improvement in knowledge of the problem and its solutions, as well as an awareness of how these solutions work [14]. Due to their complexity and the computational load they place on the system, intensive signal processing (ISP) applications require improvements in both performance and energy efficiency. Despite this, the energy efficiency barrier results in a sizable gap between the processing power of many-core systems and the computational demands of Internet service provider applications. This signals the start of a new era where improvements in transistor energy efficiency will be used as a barometer to gauge progress [15].

Physical design challenges have emerged for computer architects as fabrication technologies have progressed toward the nanometer scale. The existence of complex designs, the unpredictability of manufacturing and environmental conditions, isolated disturbances, and other factors all contribute to the decline in system precision and reliability. Researchers have recently begun to advocate for a novel approach called the Better Than Worst-Case design. This approach combines a central component that is both complex and reliable with a simple and trustworthy checker mechanism. You can build designs that can be proven to be accurate and successfully overcome the difficulties involved in deep submicron design if you grant the checker authority over the accuracy and dependability of the design [16]. MOGAC is in charge of coordinating distributed, heterogeneous embedded systems. To strike the best possible balance between cost-effectiveness and energy efficiency, a number of strict constraints are used. MOGAC achieves its objectives through the use of a communication model that combines multiple buses with point-to-point connections. Multiple Processing Elements (PEs) are the constituent parts of ASICs, and they are simulated [19].

1.1. Tackling compatibility issues

Today's technology would not exist without embedded systems. Several factors, such as safety and dependability, are impacted by embedded systems. It is essential that these systems be designed with an efficient integration of security and safety updates. The process of designing an embedded system starts with the product abstraction level, where the main goal is to make sure that hardware device implementations are both economically and environmentally friendly. In recent times, there has been a notable surge in the complexity of mobile computing devices such as smartphones and personal digital assistants (PDAs). However, researchers are focusing their efforts on enhancing both hardware and software to achieve optimal energy efficiency. This is because the necessity for batteries to be portable places restrictions on their size. Applications that rely heavily on computation have drawn the attention of an increasing number of optimization strategies in recent years.

However, given the widespread use of interactive applications, it is imperative to consider the energy efficiency of graphical user interfaces (GUIs). This study is the first of its kind to look into how a system's energy efficiency may be impacted by the way its user interface is designed. Compatibility problems often occur when new technology is integrated with older systems. It is crucial to link the hardware and software designs in order to overcome these obstacles. Embedded systems must be developed, tested, and validated throughout the development process to ensure their efficacy, reliability, and functionality. Real-time constraints must be followed during testing and validation in order for the system to be able to recognize and react to input signals in real time [15].

1.2. Methodology and Optimization Strategies

Testing and validation are crucial steps in the process of creating embedded systems because they guarantee the proper operation of the system. The methodologies for testing the hardware and software must be integrated in order to ensure the efficacy and reliability of the system. Iterative testing should be part of the validation process to find errors and functionality problems in the system. Real-time constraints must be followed during testing and validation in order for the system to be able to recognize and react to input signals in real time.

An important tactic for embedded systems is interactive design, which allows designers to test and improve concepts in real time with user feedback. Embedding testing and validation in a comprehensive manner is critical to the successful development of embedded systems. One way that designers can use their work to improve these metrics is by changing the bit widths of signals. The ability to adjust bit widths to meet the needs of specific applications is one of the biggest advantages that application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) have over instruction processors. That being said, hardware designers are finding it more and more difficult to choose the most efficient bit widths [16].

Using real-time operating systems (RTOS), which can run multiple processes in parallel without slowing down the system, is a good strategy. One option that can be taken into consideration is the use of processors and microcontrollers with lower power consumption; this will lead to a notable increase in energy efficiency and a decrease in power consumption. Making sure that the different hardware and software components are compatible with each other is crucial. Having a firm grasp of the definition and functionality of an embedded system is crucial for effectively tackling design issues that crop up during the development process.

The rapid expansion of Internet of Things applications has fundamentally changed our perception of cloud computing, requiring us to update our long-held beliefs about its capabilities [17]. Traditional central cloud computing suffers from a number of problems, including response times, network latency, and data privacy. These are but a handful of the additional problems.

As a result, the processing of data has shifted from central servers to peripheral network devices that are integrated with the Internet of Things (IoT) [18]. This is something that can be achieved by combining rigorous design and prototyping processes with testing and validation protocols. With the aid of interactive design techniques, it is possible to improve both the user experience and the interface design.

When designing vital applications that are integrated into systems, safety and security considerations must always come first. Implementing secure communication protocols, encryption techniques, and access control mechanisms are required for this to happen. It is imperative to uphold a regular schedule for the installation of updates and patches to mitigate any potential vulnerabilities or weaknesses that might arise. Proficiency in this field requires a thorough understanding of embedded systems' hardware and software components, as well as the optimization techniques and development methodologies needed for successful implementation. Developers who prioritize optimization, safety, security, and compatibility above all else when taking on design challenges can create dependable and efficient embedded systems for a diverse range of applications [19].

1.3. Exploring new optimization strategies

The ongoing evolution of embedded systems necessitates the constant need for new optimization techniques. One approach that has worked well is using machine learning algorithms to increase the power consumption and energy efficiency of real-time applications. Researching novel methods of hardware design can also help to increase the reliability and performance of embedded systems. These innovative techniques include, for instance, the use of multi-core processors and FPGA-based designs. Design space exploration, or DSE, is one of the most important areas to concentrate on when designing an embedded system. The process starts with the choice of architectural details and continues with figuring out how those details will affect the structure's overall performance. DSE must take all of these factors into account since security is becoming a more and more crucial design consideration [20].

1.4. Optimizing code for resource-constrained environments

One of the most frequent issues with embedded systems is a lack of memory or processing power. Developers must reduce the size and improve the efficiency of their code without sacrificing its functionality if they hope to successfully overcome these obstacles. To achieve these goals, techniques like code profiling, loop unrolling, and data compression can be applied.The functionality and sophistication of portable computing devices, like smartphones and personal digital assistants (PDAs), have advanced significantly. The size of the battery is determined by the customers' desire to have a portable device. Consequently, many researchers have been focusing their attention on finding ways to increase the energy efficiency of both software and hardware. Improving the efficacy, efficiency, and utilization of the resources available in embedded systems is the main goal of optimization. Numerous limitations apply to embedded systems, including those related to memory, processing speed, and energy availability. It is possible to optimize embedded systems through the application of these traditional techniques. Algorithm performance optimization aims to improve algorithmic performance in terms of efficiency and effectiveness. Use algorithms that require less time and space by making sure you choose the most efficient data structures and algorithms. By turning on the compiler optimizations, you can increase the efficiency of your code.Read this article to learn how to make the most of external storage and Flash memory.It is important to remember that the best optimization strategies for your embedded system will depend on its unique requirements and constraints. Finding a satisfactory compromise between conflicting optimization objectives is crucial, particularly when considering the unique needs of the application and the hardware itself.

1.5. Testing and validation methodologies

Strict testing and validation procedures must be used to guarantee that embedded systems function properly in practical settings. These approaches should include both functional testing, which confirms proper operation, and non-functional testing, which evaluates performance in various scenarios. Moreover, early identification of potential problems during development is made possible by interactive design approaches.

1.6. Real-time constraints

In many embedded systems that conduct their operations in real-time environments, the capacity to react rapidly is of the utmost importance. Utilizing strategies such as priority-based scheduling, interrupt-driven programming, and

pre-emptive multitasking are some of the methods that software engineers can use to remain within these constraints. Real-time constraints are an extremely important factor in embedded systems, which frequently require responses that are both easy to predict and quick to implement. Systems that operate in real time are required in order to be able to react to occurrences or stimuli that originate from the outside world within a predetermined amount of time. Regarding the real-time limitations of embedded systems, the following are a number of major considerations that should be taken into account.

1.7. Hard Real-Time vs. Soft Real-Time

The timing requirements for Hard Real-Time Systems are extremely stringent and unyielding. Failure to meet deadlines can have extremely negative consequences. The ability to respond quickly and in a timely manner are two essential components of real-time systems. When it comes to soft real-time systems, meeting deadlines is extremely important; however, it is not impossible to miss them on occasion. It is possible for persistent tardiness Sto have a detrimental effect on the performance of the system.

2. Literature Survey

The embedded system design y chart in this study shows how information in text, speech, video, audio, and graphics is becoming more and more digital.A list of essential components for embedded virtualization, which outlined the two approaches, served as the basis for the creation of the hypervisor that was presented. Hardware-assisted virtualization, security, inter-VM communication, coexistence of multiple GPOSs and real-time instances, real-time support, and direct mapped and shared devices. to expand the software development's cache size cache memory.Several examples from the literature have been practically applied to a novel multiobjective genetic algorithm that allows exploration of the Pareto-optimal set of architectures instead of giving a designer a single solution.

S. No.	Reference	Inference	Advantages	Limitations
1	Bart et al. (2002)	Cost-Effectiveness through Programmability Systematic Design Space Exploration	Cost-Effectiveness Flexibility and Reusability Adaptability	Complexity in Design and Implementation Performance Overhead
2	Mali K Sharad (2000)	Finite state machine De-multiplexer Concurrent processing and concurrent communication	Low cost Flexibility More Memory size	Combination of Hardware and software platform Low market is high
3	John N Serigo (2016)	Internet of things General purpose operting system	Real time Security Direct mapping	Memory management unit CPU bound benchmarks I/O bound benchmarks

4	Blaau W David (2005)	Dynamic voltage scaling	High performance Cost effectiveness High coverage	Testability Better than worst case design methodology
5	Austin Valeria (2000)	Typical case optimization Dynamic implementation verification architecture	Design complexity Core computation Checker slow system operation	Constraint based circuit Simple scale modeling infrastructure
6	De Y Sujit (2013)	Rate capacity effect Partial differential equations TCP/IP network interface	Recovery effect Rate capacity Life time battery	Open circuit potential Latency
7	Villa Tiziano (2001)	Cyber physical system Platform based design paradigm (PBD)	System science System engineering Bandwidth latency Heterogeneous refinement	Control system Control algorithm High level requirement
8	Dutt N (2005)	Architecture description languages Traditional hardware languages	Time to market Compilation Synthesis Content and objective	Programmable architectures Programmable embedded system
9	Racu R (2005)	System level performance Upper event function Lower event function	Real time complex Boolean function are used	Cycle internal input AND , OR are used for jitter Interrupts in lower priority task
10	S Vallerio Keith	Personal digital assistants Graphical user interfaces	Energy optimization Power reduction Performance enchancement Facilitators	Easy to learn Highly production User input cache
11	Culler E David (2001)	Wireless communication TinyOS Synchronous commands and asynchronous events	Very efficient modularity Very light wiegth threads	High level component Self organization for networking infrastucture

12	Marchal Paul (2001)	3D technologies Design automation Integrated circuit design	More effectively Reduce capacitance	Front end of line and back end of line
13	N Pedroni (2017)	high-fidelity Generate Training Data for the Meta-Mode Validate the Meta-Model	High Dimensional Computaionally Expenvise	Less Expensive Computational Model Correlation function Unexplored Critical Region
14	Ammar Manel (2016)	Electronic design automation Intensive signal processing	Energy effectiveness High speed Multi core system	Kalrays framework Static and dynamic simulation Simulation at RTL and TLM
15	Lee U Dong (2005)	Field programmable gate array Finite wordlength effetcs Optimization methods	High bandwidth To analysis the dynamic range Reduce noise	Discrete cosine transform Simulation in one dimension Design minimal area
16	Lopez Carlos Juan (2003)	Computer aided design Apllication specific instruction set processor	Control data High gain input Power consumption	Group migartion Dynamic and static programming Design quality attributes
17	Vallejo Opez (2003)	Application Specific Instruction-set Processor Worst case excution time	High performance Multiple microprocessor and microcontroller	Algorithm used for instruction set Register and bus interfacing
18	K Tha Niraj	Genetic algorithm Hardware and software cosynthesis	Multiple periodic task Real time More number of hardware and software	Point to point commication link Performance evaluation

There is a trend towards heterogeneous, distributed architectures due to embedded system optimization, which increases embedded system complexity. Multiprocessor system on chip designs (MpSoCs) integrate multiple programmable processor cores, specialised memories, and other IP components on a single chip through the use of intricate on-chip networks. In combination of hardware and software techniques which are inculde the memory based system to tackling the some issues.

3. Conclusion

Finally, we outline several different ways to build these flexible structures, one of which is the Y-chart technique. Design environments are created by architects, and by using these ideas, designers are able to carry out in-depth research on those environments. By utilizing strategies like tiling, configuring associativity, expanding the cache size, and expanding the cache line size, you can lower the number of cycles needed for data access and cache misses. We have finished the groundwork for the SymTA/S technology up to this point. Following that, we were able to add several features that enable the analysis of sophisticated embedded applications that are currently operational. This is because larger caches are able to hold more data and can also be optimized in terms of structure to improve performance when retrieving data. The experiments also looked into how the Linux operating system was affected by the overhead of the hypervisor. The hypervisor uses resources, sometimes known as "overhead" resources, to manage virtualization. These resources include CPU time and memory. The statement claims that benchmarks for both virtualized and non-virtualized executions were used to determine the overall overhead on Linux based on the outcomes of the performance tests. Despite using outdated hardware with lower processing, bandwidth, energy consumption, and storage capacity, the TinyOS approach has proven to be very effective in enabling general-purpose communication across a large number of devices. To effectively tackle these problems, a thorough plan that considers compatibility issues, difficulties in hardware and software design, and embedded system design in general must be put into action. Machine learning algorithms and other optimization techniques are crucial for addressing security issues early in the design phase and increasing energy efficiency. The ultimate goal of guaranteeing that functionality is reliable under all conditions is to apply thorough testing methodologies while carefully taking into account time constraints.

Declarations

Source of Funding

This study did not receive any grant from funding agencies in the public, commercial, or not–for–profit sectors.

Competing Interests Statement

The authors declare no competing financial, professional, or personal interests.

Consent for publication

The authors declare that they consented to the publication of this study.

References

[1] Kienhuis, B., Deprettere, E.F., van der Wolf, P., & Vissers, K. (2002). A Methodology to Design Programmable Embedded Systems. Embedded Processor Design Challenges, Pages 18–37, Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45874-3_2.

[2] Keutzer, K., Newton, A.R., Rabaey, J.M., & Sangiovanni-Vincentelli, A. (2000). System-level design: orthogonalization of concerns and platform-based design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19(2): 1523–1543. https://doi.org/10.1109/43.898830.

[3] Shiue, Wen-Tsong & Chaitali Chakrabarti (1999). Memory exploration for low power, embedded systems. In Proceedings of the 36th annual ACM/IEEE Design Automation Conference.

[4] Moratelli, Carlos, Sergio Johann & Fabiano Hessel (2016). Exploring embedded systems virtualization using MIPS virtualization module. In Proceedings of the ACM International Conference on Computing Frontiers.

[5] Austin, Todd, et al. (2005). Opportunities and challenges for better than worst-case design. In Proceedings of the 2005 Asia and South Pacific Design Automation Conference.

[6] Orgerie, Anne-Cecile, Marcos Dias de Assuncao & Laurent Lefevre (2014). A survey on techniques for improving the energy efficiency of large-scale distributed systems. ACM Computing Surveys, 46(4): 1–31.

[7] Kabilan, M., Manikandan, V., & Suresh Kumar, K. (2023). Synergizing IoT, IoE, GSM Technology, and Deep Learning Models for Advanced Security Applications: A Comprehensive Overview. Irish Interdisciplinary Journal of Science & Research, 7(4): 38–46.

[8] Nuzzo, Pierluigi, et al. (2015). A platform-based design methodology with contracts and related tor the design of cyber-physical systems. In Proceedings of the IEEE, Pages 2104–2132.

[9] Mishra Prabhat & Nikil Dutt (2005). Architecture description languages for programmable embedded systems. In IEEE Proceedings-Computers and Digital Techniques, Pages 285–297.

[10] Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., & Ernst, R. (2005). System level performance analysis–the SymTA/S approach. In IEEE Proceedings-Computers and Digital Techniques, Pages 148–166.

[11] Vallerio Keith, S., Lin, Z., & Niraj, K.J. (2006). Energy-efficient graphical user interface design. IEEE Transactions on Mobile Computing, 5(7): 846–859.

[12] Culler David, E., et al. (2001). A network-centric approach to embedded software for tiny devices. Embedded Software: First International Workshop, EMSOFT 2001 Tahoe City, CA, USA.

[13] Marchal, P., Bougard, B., Katti, G., Stucchi, M., Dehaene, W., Papanikolaou, A., & Beyne, E. (2009). 3-D technology assessment: Path-finding the technology/design sweet-spot. In Proceedings of the IEEE, Pages 96–107.

[14] Turati, P., Pedroni, N., & Zio, E. (2017). Simulation-based exploration of high-dimensional system models for identifying unexpected events. Reliability Engineering & System Safety, 165: 317–330.

[15] Ammar, Manel, Mouna Baklouti & Mohamed Abid (2016). The performance-energy tradeoff in embedded systems design: A survey of existing design space exploration tools and trends. International Journal of Computer Science and Information Security, 14(5): 381–391.

[16] Lee, D.U., et al. (2006). Accuracy-guaranteed bit-width optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(10): 1990–2000.

[17] López-Vallejo Marisa & Juan Carlos López (2003). On the hardware-software partitioning problem: System modeling and partitioning techniques. ACM Transactions on Design Automation of Electronic Systems, 8(3): 269–297.

[18] Ayari, R. (2018). Optimization and Mining Methods for Effective Real-Time Embedded Systems. Doctoral Dissertation, Ecole Polytechnique, Montreal, Canada.

[19] Dick Robert P., & Niraj, K.J. (1998). MOGAC: a multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 17(10): 920–935.

[20] Joseph Abin John, Nidhin Sani, Suresh Kumar, K., Ananth Kumar, T., & Nishanth, R. (2022). Towards a novel and efficient public key management for peer-peer security in wireless ad-hoc/sensor networks. In 2022 International Conference on Smart Technologies and Systems for Next Generation Computing, IEEE, Pages 1–4.