ISSN 2250-3153

An Adaptive Framework for the Selection of Embedded Operating Systems

Paripati Lohith Kumar
kumarlohith12@yahoo.com

Abstract- The embedded system area today is faced with the challenge of implementing the applications that can execute efficiently on limited resources and that meet nonfunctional requirements such as timeliness, robustness, dependability and performance. At the core, these requirements need to be managed by the operating system. Thus, an embedded operating system has to address not only the aforementioned issues but also the issues arising out of the requirements of newer embedded applications. Such types of constraints are difficult to meet in an embedded operating system and this is the reason that the contemporary embedded operating systems are designed for specific application areas. This research work aims at identifying a common and adaptive framework for embedded operating systems so that a customized embedded operating system can be generated according to the requirements of an application.

Index Terms- embedded system, RTOS, embedded system design.

I. INTRODUCTION

Embedded systems can be defined as computing systems with tightly coupled hardware and software that are designed to perform a dedicated function. The word embedded reflects the fact that these systems are usually an integral part of a larger system.

It is not hard to find a large variety of applications where embedded systems play an important role, from small stand-alone systems, like a network router, to complex embedded systems supporting several operating execution environments as can be found in avionic applications. This variety of applications also implies that the properties, platforms, and techniques on which embedded systems are based can be very different. The hardware needs can sometimes be achieved with the use of general purpose processors, but in many systems specific processors are required, for instance, specific digital-signal-processing devices to perform fast signal processing. Memory management capabilities are necessary in some systems to provide memory protection and virtual memory. Special purpose interfaces are also needed to support a variety of external peripheral devices, energy consumption control, and so on.

Nowadays, the use of processor-based devices has increased dramatically for most of the daily activities, both professional and leisure. Mobile phones and PDAs are used extensively. Consumer electronics (set-top boxes, TVs, DVD players, etc.) have incorporated microprocessors as a core system component, instead of using specific hardware. This trend is expected to grow exponentially in the near future.

A. Research issues in embedded operating systems

Embedded applications are characterized by the following common features which also indicate the constraints to be managed by the embedded operating system.

1. **Limited resources:** There are often strong limitations regarding available resources. Mainly due to cost and size constraints related to mass production and strong industrial competition, the system resources as CPU, memory, devices have been designed to meet these requirements. As a result of these limitations, the system has to deal with an efficient use of the computational resources.
2. **Real-time application requirements:** Some of the applications to be run in these devices have temporal requirements. These applications are related with process control, multimedia processing, instrumentation, and so on, where the system has to act within a specified interval.
3. **Embedded control systems:** Most of the embedded systems perform control activities involving input data acquisition (sensing) and output delivery (actuation). Deterministic communications are also another important issue.
4. **Quality of service:** Feedback based approaches are being used to adjust the performance or quality of service of the applications as a function of the available resources.
5. **Rapid development and deployment:** Due to fierce competition in embedded systems applications the design time, prototyping time and deployment time play a critical role as these timings decide the time-to-market.

The challenge is how to design and implement applications that can execute efficiently on limited resource and that meet nonfunctional requirements such as timeliness, robustness, dependability and performance. Although these requirements belong to the embedded applications they need to be managed by the operating system. Thus, an embedded operating system needs to address not only the aforementioned issues but also the issues arising out of the requirements of newer embedded applications. Such types of constraints are difficult to meet in an embedded operating system and this is the reason that the embedded operating systems are designed for specific application areas. In this paper we have proposed a common and adaptive framework for embedded operating systems, so that a customized embedded operating system can be generated according to the requirements of an application.

This paper is organized as follows: in section 2 we survey the related work in selecting the RTOS and reconfiguration of it. Section 3 describes the important parameters of Embedded operating Systems and its role in embedded system design. In Section 4 we explain our idea of and adaptive framework for the selection of embedded operating systems. Section 5 provides the conclusion and direction of future work.

II. RELATED WORK

In this section, a survey of related works in the area of reconfigurable real time operating systems is presented. The past decade has seen a significant research work on selecting the RTOS. Designers are impressive task when selecting the RTOS for specific applications like Space, Security, military, process industry, communications, robotics, Data Acquisition, consumer electronics and so on in which each application demands specific requirements.

Embedded operating systems allow you to develop applications faster. They can require a little more overhead, but as the technology improves, the overhead seems to diminish. In Greg Hawley, he has provided criteria for selection of RTOS based on the processor and based on the requirements. He also considered many other parameters like, company profile, licensing policy technical support etc.

In a selection methodology for the RTOS market various method are adopted for space applications. This paper describes the elimination criteria for selection of RTOS to a very specific space application and ranked the existing commercial RTOS that are available in the market but they have not provided the generic framework for RTOS selection. In [20], how to select your RTOS described the framework for selection of RTOS for a class of applications and its characteristics that meets the application but it doesn't provided the methodology to select the RTOS based on the designers/developers requirements which are incorporated in this paper. Criteria for selection of a RTOS need to be much more flexible and much less specific.

Since 1940, several optimization problems have not been tackled by classical procedures including: Linear Programming, Transportation, Assignment, Nonlinear Programming, Dynamic Programming, Inventory, Queuing, Replacement, Scheduling etc. Embedded systems are special-purpose systems. They are often designed to perform very specialized tasks. Any operating system running on such a specialized system could benefit greatly from adapting to specific requirements. Therefore, configurable operating systems seem advantageous for embedded systems. The requirements for a dynamic configuration system for embedded operating systems are as follows
(i)The system should allow low-level resource managers to be configured to allow maximum flexibility. (ii)The run-time overhead should be minimal. (iii)The memory footprint should be small. (iv)The system should not require a hardware memory management unit. (v)Re-configuration should be reasonably fast compared to the lifetime of an application. (vi)Real-time computing should be possible in between re-configurations.

It has been shown that embedded operating systems can be configured dynamically by loading and linking code into the system at run-time.

A lot of work has been produced in the domain of adaptive architectures. Different techniques have been introduced for clock and voltage scaling [8], cache resources [5], and functional units [9] allocations. These approaches can be classified in the category of local configurations based on specific aspects. Real Time Operating System (RTOS) for hardware management has been recently introduced. Proofs of concepts are exhibited in [9, 4]. These experiments show that RTOS level management of reconfigurable architectures can be considered as being available from a research perspective. In [7], the RTOS is mainly dedicated to the management (placement/communication) of hardware tasks. Run-time scheduling of hardware tasks/algorithms are described in [4]. In [9], the scheduling layer is an OS extension that abstracts the task implementation in hardware or software; the main contribution of this work is the communication API based on message passing where communication between hardware and software tasks is handled with a hardware abstraction layer. Moreover, the heterogeneous context switch issue is solved by defining switching points in the data flow graph, but no computation details are given. In [6], abstraction of the processor and programmable hardware component boundary is supported by hardware thread interface concept, and specific real time operating system services are implemented in hardware. In [10, 12] service oriented architecture is discussed for the design and development of embedded applications. The current state of the art shows that adaptive, reconfigurable, and programmable architectures are already available, but there is no real complete solution proposed to guarantee safe configuration of operating system with reference to application requirements. It motivates for the investigation of an adaptive framework for embedded operating system generated for application specific requirements.

III. SELECTION OF RTOS

Ranking RTOS is a tricky and difficult because there are so many good choices are available in the market . The developer can choose either commercial RTOS (44% developers are using) or open- source RTOS (20) or internally developed RTOS (17 %).This shows that almost 70% of developers are using the RTOS for their current projects and are migrating from one RTOS to another due to various reasons. To handle the current requirements of the customers, developers are using 32 bit controllers in their projects in which 92% projects/ products are using RTOS and 50% of developers are migrating to another RTOS for their next project. This influences importance of the selection of right RTOS to a particular project so that it meets all the requirements and fulfills its intended task.

In all of the related work authors have used the elimination criteria which are manual and it takes more time and need the detailed specifications of all the existing commercial RTOS's. In order to select RTOS, the designer first identify the parameters for selection based on the application and the intended requirements are provided to the systems through an interactive user friendly GUI.

A. Important embedded operating system parameters

Among the different parameters for selecting the RTOS, the ones used in our system are: 1. Interrupt Latency, 2. Context switching 3. Inter task Communication (Message Queue mechanism, Signal Mechanism, Semaphores), 4.Power management (Sleep mode, Low power mode, idle mode, Standby mode) 5. No. of Interrupt levels 6. Kernel Size 7.Scheduling

Algorithms (Round Robin Scheduling, First Come First Serve, Shortest Job First, Preemptive Scheduling etc), 8.Interrupt Levels, 9. Maintenance Fee 10 Timers 11. Priority Levels 12.Kernel Synchronization (timers, mutexes, events, semaphores etc), 13. Cost, 14. Development host, 15.Task switching time and 16 Royalty Fee. There are more parameters like target processor support, Languages supported, Technical support etc. are also important which are considered by the developer. We have used the knowledge base for storing the features of an embedded operating systems Then we use our framework for selecting embedded operating system, which is described in the following section. Our system will output a set of EOS from which one will be selected by considering the processor support, languages supported and Technical Support etc which are also important.

IV. PROPOSED FRAMEWORK

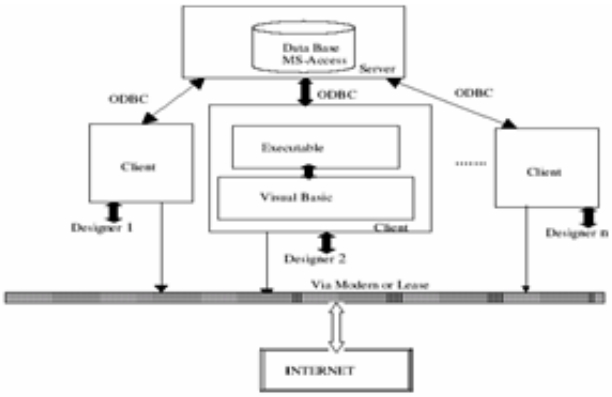

Fig. 1 Architecture of the System

We have developed a graphical user interface so that the user can specify the weights for the parameters of the RTOS for his application. The parameters specified by the user using the GUI are given below.

Development Methodology – Cross Weight – 1
RTOS Supplied as – Object Weight – 2
Development Host – UNIX Weight – 3
Standard – POSIX .1 Weight – 4
Kernel ROM – 280K/4M Weight – 5
Kernel RAM – 500K/4G Weight – 6
Priority Levels – 512 Weight – 7
Multi process Support – No Weight – 8
Multiprocessor Support – No Weight – 9
MMU Support – No Weight – 10
Royalty free – No Weight– 11
Standard phone support – Paid Weight – 12
Preferred phone support – Paid Weight – 13
Base price – 7495$ Weight – 14
Maintenance fee – 15% of list price Weight– 15
Task switching time – 4us to 19us Weight – 16

A. Fitness function

The fitness function is the weighted sum of the parameters given in section 3, each of which contribute the "goodness" of the final selection of embedded operating system. Fitness is s evaluated by using the fitness function (FF) which is given by

$FF = \sum (W_i F_i)$

Where W_i is the wait of the i^{th} parameter and F_i is the fitness value of the i^{th} parameter.

Let us first consider the weights. Each application of an embedded system will have specialized requirements. The requirements can be characterized using the parameters specified in section 4 by assigning appropriate weights. The weights change depending on the application. For example, for children toys, cost may be the main criteria and hence will have maximum weight while for robotic applications response time would be the parameter with maximum weight. To meet these specifications, the user has to specify the weights for each parameter so that an appropriate OS will be selected. In the fitness function, Wi is the weights assigned by the user. consider, now, the fitness values. The parameters of RTOS given above have different values for different RTOS. For example, the interrupt latency can be 5ns for one RTOS and 15ns for another. The different values are mapped to a scale and the value on the scale is the fitness value. For example, if the scale for interrupt latency is 5 to 15 then, for the RTOS with 5ns as interrupt latency, the fitness value is 1 as it is better to have low interrupt latency. Since the values of these parameters are available beforehand for the RTOS that are available in the market, the fitness values are precompiled at the time of generating the database of RTOS. However, the designer can alter the values if needed. Now, by using the fitness function FF defined earlier, we evaluate the overall fitness value for the given criteria.

The user gets an appropriate RTOS just by giving the specifications and the desired accuracy and the whole search based on those specifications is carried out by the system and hence the result is provided through an easy designed interactive GUI. The user has the option of specifying the accuracy percentage to carry out his search which could vary depending on the level of strictness required, which is an efficient method compared to other methods which uses the elimination criteria The user has the provision of selecting more than one option in each parameter thus making his search more advanced in terms of parameters. Choosing the most appropriate RTOS can still result in significant cost savings, improved level of technical support and high levels of product integration.

V. CONCLUSION

Embedded applications have become a popular these days due to the complexity in the system. To meet those complexities, the developers are given the invariable task of making the embedded software. There are quite large number of embedded operating systems are available in the market and one dose get confused as to which one such that it provides the efficient embedded systems design in terms of cost, power consumption, reliability, speed etc. In this paper, we described a Simple adaptive framework that is designed to find the suitable embedded operating systems for a specific application. The methodology described for RTOS selection is unique and efficient for large number of RTOS's. It has user-friendly graphical interface (GUI) though which the designer can alter the specifications and specify the new requirements for embedded operating system selection for a given application. It generates the optimal RTOS based on the requirements that are entered by the user keeping in mind the amount of accuracy required. This is done with the help

of fitness function. Our analysis and the developed system gives the user a portal to decide a real time operating system which most suits his choice of parameters and is the most optimal one available for that purpose. The designer has an option of choosing from pre-defined input or can specify his/ her own input.

REFERENCES

[1] RTOS for FPGA-A White paper by Colin Walls, Accelerated Technology, Embedded Systems Division, Mentor Graphics Corporation (www.mentor.com/fpga).

[2] Victor Yodaiken and Michael Barabanov "design Document about RTLinux in FSMLabs 1997.

[3] Koza, John R., Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press, 1992.

[4] Cramer, Nichael Lynn: "A Representation for the Adaptive Generation of Simple Sequential Programs" Proceedings, International Conference on Genetic Algo, July 1985 [CMU], pp183-187.

[5] Wayne Wolf, Computers as Components: Principles of Embedded Computing System Design, Morgan Kaufmann Publishers, 2001.

[6] Frank Vahid and Tony Givargis, Embedded System Design: A Unified hardware/ Software introduction, John Wiley& Sons, 2002.

[7] Scott Rosenthal, Selecting an embedded Processor involves both simple and non-technical criteria, June, 1997.

[8] Sharad Agarwal, Edward Chan, Ben Liblit, Processor Characteristic Selection for Embedded Applications via Genetic Algorithms, December,1998.

[9] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[10] Khawar M. Zuberi, Kang G. Shin, (2001). EMERALALDS: A Small-memory Real Time Microkernel", IEEE Trans on Software Eng.; Vol.27, No.10, Oct.pp. 909 - 929.

[11] Shanil Mechant, Kalen Dedhia,"Performance computation of RTOS, thesis, Dept of electrical Eng., Columbia University.

[12] Mehrdal Dianati, Insop Song, Mark Treiber, "An Introduction to Genetic Algorithms and Evalution Stragies" Univ. of Waterloo, Canada.

[13] Martin Timmerman, "RTOS Evaluations", Real Time Magazine 98(3) March 1998.

[14] David Stewart, "Measuring Execution time and Real Time Performance", Embedded Systems conference, San Francisco, April 2001.

[15] K. Obenland, " Real Time Performance of Standards based Commercial Operating Systems, Embedded Systems conference, San Francisco, April 2001.

[16] Philip Melanson, Siamak Tafazoli, "A selection methodology for the RTOS market", DASIA 2003 conference, Prague Czec Republic, June 2003,.

[17] Ger Scoeber, "How to select your RTOS" Bits and Cips Micro-event: Embedded operating Systems, Jan 29th 2004.

[18] Greg Hawley, "Selecting a Real-Time Operating System" Embedded Systems Programming Magazine. [www.embedded.com]

[19] Ljerka Beus-Dukic, "Criteria for Selection of a COTS Real-Time Operating System: a urvey"[ljerka.beus@unn.ac.uk]

[20] Jim Turly, "Embedded Systems survey: Operating systems up for grabs", Embedded Systems Design, May 24 2005.