INTERNATIONAL JOURNAL FOR TRENDS IN ENGINEERING & TECHNOLOGY
VOLUME 2 ISSUE 1 – OCTOBER 2014 - ISSN: 2349 - 9303

Non Preemptive Scheduling based Power/Energy Management and Memory Partitioning Using RTOS

K.S.Aswathrangaraj [1]
[1]RVSCET, Embedded System Technology,
aswathrangaraj@gmail.com

L.Senthilmurugan[2]
[2]RVSCET, Assistant Professor, EEE
Senthil2010@gmail.com

Abstract— Power and energy have ended up progressively critical concerns in the outline and execution of today's multi core chips. Picking the right controller and advancement environment for applications that need to be backing numerous years or even many years of operation of a battery is not simple. The proposed framework is non preemptive priority scheduling, which exploits regularly overlooked element execution information, so as to diminish power utilization by in excess of 20 percent with a noteworthy increment in execution. The results indicate a power saving, as well as a critical change in the execution, execution for every watt, and execution time watt (energy) for a task.

Numerous installed frameworks utilize programming over seeing memories known as Scratch-Pad memories (SPM). SPM is programming controlled and subsequently the execution time of uses for such frameworks could be precisely anticipated. Programming the task of an inserted application of the processors and partitioning the accessible SPM plan among these processors are two discriminating issues in such frameworks. Regularly, these are considered independently; such a decoupled methodology may miss better quality timetables. In this paper, it exhibits an incorporated methodology to assignment programming and SPM partitioning to further lessen the execution time of implanted applications.

Index Terms— RTOS, dynamic process priority, memory partitioning, task scheduling, scratch pad.

1 INTRODUCTION

With current controller getting to be constantly controlled cognizant and fusing different power, sparing modes of operation, ARM Cortex-M3, broadly received for controller outlines, and utilizes the absolute most progressive power sparing technology seen among 32-bit processors. Vitality Micros Cortex-M3-based LPC2148 includes low-control peripherals and offers a scope of power sparing modes for different operating circumstances. The business in general is concentrating on power protection, and the LPC2148 building to design can serve as a sample of industry patterns of this directive. To convey genuine vitality benevolent products, controller producers need to consider a few components, and with a limited measure of charge accessible from a battery cell, it is the way the controller unit utilizes vitality control about whether that has the effect [1]. Examining inserted applications demonstrate that numerous frameworks use up to 99% of their time sitting tight for a pin change, a clock to match a contrast esteem or with get information. Taking a gander at these holds up states where the CPU could be resting, it gets to be obvious that minimizing the power and time is as paramount amid slumber as in dynamic periods.

In a continuous framework, a periodic task could be existed on one of three essential states: running state, prepared state and blocked state. At the point when a periodic task is done, its state will be adjusted to the running state to the blocked state. At that point, it will be woken up (i.e. changing to the prepared state) after its period. The scheduler experiences all blocked undertakings and chooses when to awaken assignment and move it to prepared rundown [2]. To just perform this, the scheduler utilizes a variable to deal with the following woken up time of assignment. This variable will be checked at each framework tick to choose whether the assignment is woken up or kept in the blocked state. Notwithstanding, not all periodic task runs for the force on time to the end of battery time.

The majority of intermittent undertakings is dynamic for a time of time and afterward be hindered until they are enacted by an outer occasion. The Case of points, the screen of a cellular telephone does not have to be redesigned regularly in standby modes. Overseeing productively occasional undertakings will decrease the framework time and subsequently diminish power utilization. On the off chance that intermittent tasks are effectively tasks utilizing to need scheduled, the processor vitality utilization continuously frameworks could be decreased.

While installed frameworks get to be progressively mind boggling, the increment in memory access speed has neglected to stay aware of the increment in processor speed. This makes the memory access inactivity a significant issue in booking inserted applications of implanted frameworks. To mitigate such issues, numerous advanced controller frameworks use programming controlled memories known as scratch pad memories (SPMs), which permit execution times to be anticipated with precision [3]. The calculation time of a system on a processor relies on upon measures of SPM are designated to that controller/processor for executing a task.

2 LITERATURE SURVEY

Power-aware processing has ended up well known as of late and numerous methods have been proposed to oversee processor vitality utilization for conventional ongoing applications. One such strategies, including element recurrence scaling, can bring down a center's recurrence amid the execution of a non-CPU escalated task, accordingly bringing down execution. Gang Zeng, 2008 proposed systems to minimize the force utilization of unmoving state by considering the occasional intrude on administrations [4]. Gruian, 2001 presented a hard constant tasks planning with settled needs appointed. It might be directed at presumption of free undertaking set [5]. An alternate methodology is presented in investigation into which exhibited that ease off variables are connected to synchronize getting to of imparted assets to lessen power utilization [6].

Numerous examination gatherings have concentrated on the issue of undertaking booking of uses of various processors where the destination is to minimize the execution time. Benin tackled the planning issue utilizing imperative programming and the memory dividing issues utilizing numbers directs programming. Ahmed introduced a correlation among calculations for planning task diagrams onto a set of homogeneous processors on a differing set of benchmarks to give a reasonable assessment of every heuristic focused around a set of presumptions. De Michele concentrated on the mapping and booking issue onto a set of handling components as a fittings/programming Co-design. Neumann and Mar Weddell utilized number programming to understand the fittings/programming co design apportioning issue. Cho proposed a precise planning model of equipment/programming correspondence building to design to enhance timing precision [7]. Angiolini ideally tackled the issue of mapping memory areas to SPM areas utilizing an element programming.

3 METHODOLOGY

3.1 Software Design

The proposed non Preemptive priority scheduling for Power (Energy management) and memory partitioning using RTOS in ARM is shown in fig 1 block diagram.

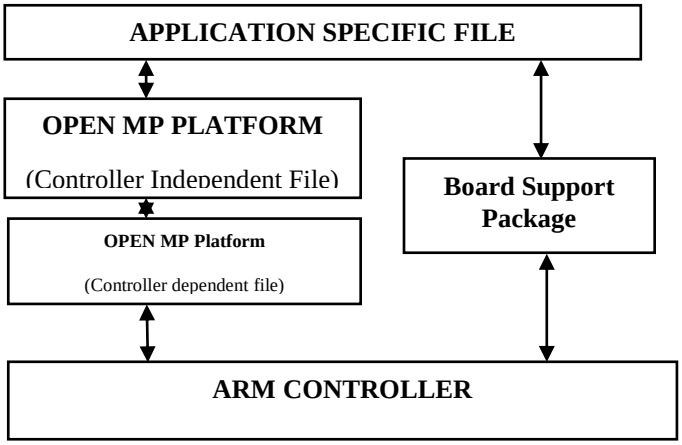

Fig 1. Block Diagram

Selection of RTOS

The proposed system uses UBUNTU 12.0 Linux based Real Time operating system in practical design for its increased security by tool, default character encoding and compiles packages using G CC [8].

The UBUNTU 12.0 OS supports Open MP (Open Multi processing) platform which is an API support shared multiprocessing program in C, C++ and FORTRAN. By Open MP, it can achieve both task and data parallelism. By default the Open MP codes are visible to all threads in a shared programming model [9]. The core element of Open MP is shown in fig 2 as follows.

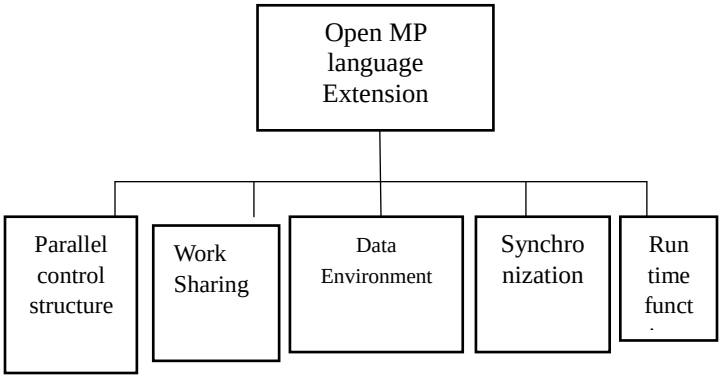

Fig 2. Core Element of Open MP

The core element of Open MP consist of thread creation, sharing of works, data environment management, environment variables and user level run time routines. Open MP uses in C, C++ programs.

a. Application selection file

This layer cover program files of task in a framework. For example UART, ADC, DAC, GSM, LCD, RTC configuration.

b. Controller independent files

This covers OS service files such as shown in above core elements and clauses for data sharing, synchronization and scheduling.

c. Controller dependent files

This files contains OS based CPU files which covers information related to target controller. The main work of this files are to transplant the INCLUDE.H files to the target controller.

d. Board support package

It is a software which allows access from OS and application software to the target controller hardware. The main responsibilities is to initialize the hardware and provide a start point for the RTOS.

3.2 Software Approach

An RTOS gives the engineer a skeleton on which to assemble and sort out the peculiarities of the framework. The tool kit that goes hand in hand with the RTOS ought to additionally give administrations, for example, between task correspondence and time administration. Actually for frameworks that have no need of the constant capacity, the code might be cleaner and better composed if focused around a RTOS, incompletely on the grounds that it pushes code reuse. The coordination of a RTOS can tackle a mixed bag of issues that can happen in application code, since it gives multitasking ability and permits the application to be broken down into more modest pieces. Each one undertaking is relegated it need focused around its critics, and Preemptive scheduling guarantees that the controller runs the assignment that has the most noteworthy need among those that are prepared to-run. By and large, including a lower need task won't influence the responsiveness of the framework to high need undertakings.

The Acknowledgment of Programming Software

At the point when RTOS is effectively transplanted to the protested processor, then it ought to compose the application programming of the microcomputer protection for further steps.

1. Making of Tasks

Open MP platform can oversee multi parallel handling tasks. The four most noteworthy need assignments and the four least need undertakings are reserved for the framework. The application programming first ought to make the task.

2. The partition of tasks and the dissemination of priority

The standard of task partition is minimizing the coupling in the middle of tasks and makes the capacity plainly under the considering of framework constant and the software efficiency.

3. **The Scheduling and Exchanging of the task**

Open MP is continually working the prepared most astounding priority task. The planning of task class firstly are working the current prepared most astounding priority tasks. At the point when the current task hang-up by the delay time, do the assignment switch and execute the higher priority task. At the point when the deferral time is to, and the hanged-up assignment is in a prepared state, and afterward do the task scheduling once more.

4. The Synchronization and Communication of Task

The correspondence between tasks can utilize semaphore, message mail box, and message queue or event flag gathering. The semaphore and the event flag gathering to finish the conduct synchronization. The message mailbox and the message queue can finish the conduct synchronization as well as can finish the data communication.

5. Acknowledgment Methodology of Tasks

The acknowledgment of the task is likewise an executing stream of the assignment. To the assurance assignment of the microcomputer security gadget firstly is through the clock to cause the timing intrusion and the interrupt service routine sends the protection semaphore. The most priority task scheduling which holding up this semaphore gets the semaphore. The task peruses the most recent examining worth structure the testing cycle stockpiling and calls the Fourier calculation to process the electric quantity amplitude and the phase angle. And after that calls activity foundation capacity to judge whether act or alert. In the event that activity or alert is required, compose the transfer control word. In the wake of completing this tasks it additionally can call showcase tasks and print task to finish shortcoming recorder.

4 CONCLUSION AND FUTURE WORK

As a typical methodology to enhance the execution and to spare power utilization of real time embedded system, the processor must stay in low power modes to the extent that this would be possible and the time for the scheduler to deal with all tasks must be as low as could reasonably be expected. Overseeing productively of periodic tasks will let processor stay in low power modes longer and spare time for task management. The memory partitioning of embedded applications is efficient based on task using non Preemptive priority scheduling with scratch pad memory. Thus power/energy is managed and memory partition based on priority scheduling results in high efficient than other scheduling algorithm.

In Future work the RTOS is ported into ARM controller of LPC2148 using IAR embedded workbench software in order to perform the task in the framework of based on Non Preemptive priority algorithms.

REFERENCES

[1] “ARM Processor Instruction Set Architecture” archived from the original on 15 April 2009. Retrieved 18 April 2009.
[2] Vu DIN H-DUC, “An efficient scheduling for low power in Real Time Embedded Systems”. International conference on Advanced Technologies for Communications (ATC2012).
[3] Hassan and Ramanujan, “An Effective solution to Task scheduling and Memory partitioning for multiprocessor system-on-chip”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, VOL. 31, NO. 5, May 2012.
[4] Gang Zeng and Takada 2008. Dynamic Power Management for Embedded System Idle state in Presence of Periodic Interrupt Services, Information and Media Technologies (2008) 661-670.
[5] Flavius Gruian 2001. “Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors”. Low Power Electronics and Design International Symposium,

Huntington Beach, CA, USA, Aug, 6-7, 2001, pp. 46-51.
[6] Gupta 2006. "Energy aware task scheduling with task synchronization for embedded real time systems". Computer-Aided Design of Integrated Circuits and Systems IEEE Transactions, 25 (6) (2006) 1024-1037.
[7] Cho and K.Choi, "Scheduling with accurate communication delay model and scheduler implementation for multiprocessor system-on-chip," Des. Auto-mat. Embedded System., vol.11, nos. 2-3, pp. 167-191, 2007.
[8] Wikimedia Foundation, Inc., 2011. [Online]. Available: WWW.Wikipedia.org /wiki/ Ubuntu
[9] Wikimedia Foundation, Inc., 2011. [Online]. Available: WWW.Wikipedia.org /wiki/ Open MP

K.S.Aswathrangaraj received the B.E degree in Electronics and Communication Engineering from KPRIET, Coimbatore, India, in 2013.
He is currently pursuing M.E degree in Embedded System Technologies at RVSCET, Coimbatore, India. His current interest include embedded system.

L.Senthilmurugan received the B.E degree in EEE from Park College of Engineering, Coimbatore, India, in 2006, and the M.E degree in Embedded System Technologies from Anna University Coimbatore, India, in 2009.
He is currently working as an Assistant Professor with Department of EEE. He is pursuing his PhD degree at Anna University Chennai, India. His current interest include embedded system, electric drives and control.