CPU Scheduling for Power/Energy Management Using RMS Algorithm

G.Dhivya, A.SenthilKumar

Abstract— **Power and energy have ended up progressively critical concerns in the outline and execution of today's multicore/numerous core chips. One problem dealing with power/energy management for heterogeneous multicore processors is: Given a set of processes, each having identical default priorities, in a given task to be executed by a heterogeneous multicore/many core processor system, schedule each process in this task to execute upon the CPU(s) in this system such that the global power budget is minimized, the performance loss of all processes is minimized. The results indicate performance of a power (energy/execution time) saving, as well as a critical change in the execution, execution for every watt, and execution time watt (energy) for a task using Priority algorithm.**

***Index Terms*— CPU, Heterogeneous, Priority, Multicore, Numerous Core.**

I. INTRODUCTION

Multi-core processors have become a primary trend in the current processor development. As a result, future high-performance real-time embedded systems are anticipated to be equipped with multi-core processors, or even many-core processors. This increased power consumption creates heat dissipation, which leads to higher costs for air conditioning, thermal packaging, fans, and electricity. Increased heat and temperature of hardware, such as the CPU, can also lead to decreased longevity and greater incidence of failure of these components. Also, the integration and design of a multicore chip is more difficult to manage, than a lower-density, single-chip design due to thermal constraints. Heterogeneous architectures may achieve a higher performance per watt than comparable homogeneous systems due to the ability of each application to run on a core that best suits its architectural properties. Real-time embedded systems are designed according to maximum execution times. In order to keep the carefully balanced timing behavior untouched the processor resource is statically allocated. The calculation time of tasks, especially in driver assistance systems and systems for highly automated flying, is increasingly dependent on the situation. Even though architectures with cores operating at different frequencies may still be considered homogeneous, such architecture may contain cores specialized for certain tasks. Context switching is less CPU intensive processes that benefit less from executing upon higher frequency cores. In previous method used priority scheduling algorithm. In this method the low priority task has been removed to avoid these RMS algorithm used. It has no resource sharing and having static priorities.

Manuscript received March 12, 2015.

***G. DHIVYA**, M.E. Embedded System Technologies, Department of ECE, Anna University, Sri Shakthi Institute of Engineering and Technology, Coimbatore, India.*

***A.SENTHIKUMAR**, Department of Electronics and Communication, Sri Shakthi Institute of Engineering and Technology Coimbatore, India.*

II. EXISTING SYSTEM

A global power manager that attempted to meets a specific global power budget by adjusting the power modes of individual cores. They assigned one of three power modes to each core, Turbo, Efficient1, and Efficient2. However, in contrast to our work, the authors used a simulator to evaluate their work, rather than real hardware. Also, they assumed a per-core DVFS (dynamic voltage and frequency scaling) knob to be available to their global power manager, unlike our work, which uses identical cores operating at different but fixed frequencies.

To minimize the power consumption of idle state, power management therefore should consider the effect of periodic interrupt services. In case the periodic interrupt cannot be disabled, we formulate the power consumption of idle state, and propose static and dynamic approaches for the optimal frequency selection to save idle power. On the other hand, in case the periodic interrupt can be disabled, the configurable clock tick to disable the interrupt service until the next task is released so that the processor can stay in the low power mode for longer time. In their model, tasks were assigned permanently to processors (partitioned scheduling) and were assigned rate-monotonic priorities that were inversely proportional to their periods. They measured the efficiency of their algorithm in terms of both total energy consumption and feasibility. They proved that this problem is NP-Hard in the strong sense on m greater than or equal to 2 processors, even when feasibility is guaranteed a priori. However, unlike our algorithm, they used DVS (dynamic voltage scaling) to scale the frequency of their CPUs and used a simulator to measure their results.

.

III. PROPOSED SYSTEM

In proposed work implement RMS algorithm instead of Priority Scheduling Algorithm.

RMS CPU SCHEDULING

In computer science, rate-monotonic scheduling (RMS) is a scheduling algorithm used in real-time operating systems (RTOS) with a static-priority scheduling class. It has the following feature:

ISSN: 2278 – 909X

- No resource sharing
- Deterministic deadlines are exactly equal to periods
- Static priorities
- Static priorities assigned according to the rate monotonic conventions
- Context switch times and other thread operations are free and have no impact on the model

Tasks are periodic => constant intervals between requests

Each task must be completed before the next request for it occurs and the tasks are independent then it request for a certain task does not depend on the initiation/completion of requests for other tasks Run-time of each task is constant.

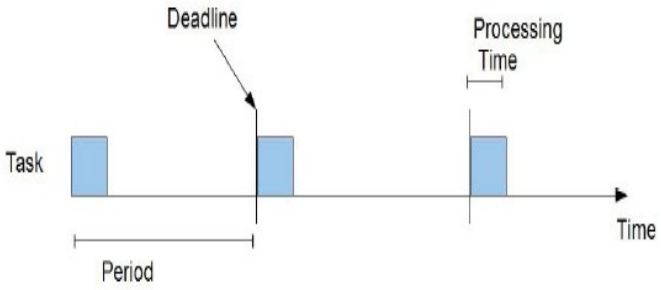

Fig 1: RMS Algorithm

The CPU utilization is below a specific bound (depending on the number of tasks). The schedulability test for RMS is:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

Where,

C_i is the computation time,

T_i is the release period (with deadline one period later),

n is the number of processes to be scheduled. For example, $U \leq 0.8284$ for two processes. When the number of processes tends towards infinity, this expression will tend towards:

$$\lim_{n\to\infty} n(\sqrt[n]{2} - 1) = \ln 2 \approx 0.693147\ldots$$

The rate-monotonic priority assignment is *optimal*, meaning that if any static-priority scheduling algorithm can meet all the deadlines, then the rate-monotonic algorithm can too. The deadline-monotonic scheduling algorithm is also optimal with equal periods and deadlines, in fact in this case the algorithms are identical; in addition, deadline monotonic scheduling is optimal when deadlines are less than periods.

For the task model in which deadlines can be greater than periods, Audsley's algorithm endowed with an exact schedulability test for this model finds an optimal priority assignment.

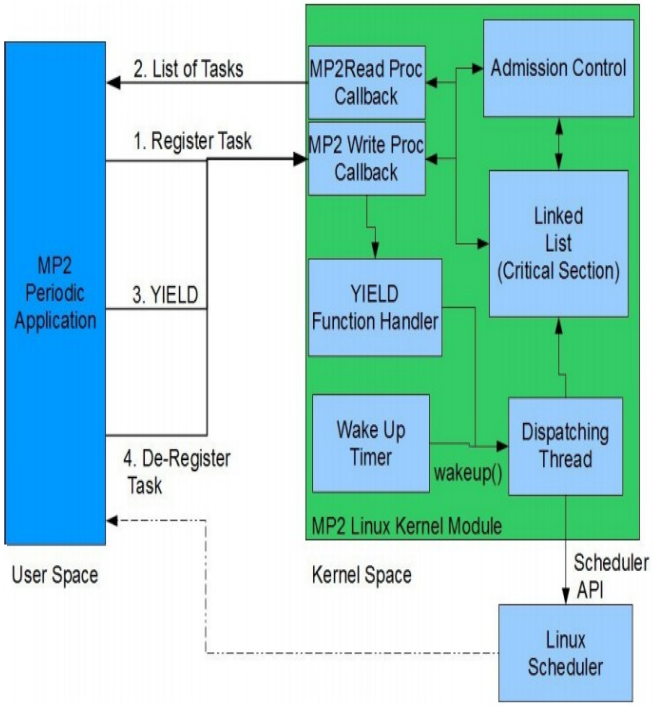

Fig 2: Overview of the Process

OPEN MP

OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran,[4] on most processor architectures and operating systems, including Solaris, AIX, HP-UX, Linux, Mac OS X, and Windows platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer. OpenMP is an Application Program Interface (API), jointly defined by a group of major computer hardware and software vendors. OpenMP provides a portable, scalable model for developers of shared memory parallel applications.

The API supports C/C++ and Fortran on a wide variety of architectures. Runtime library functions and environment variables are also covered. This tutorial includes both C and Fortran example codes and a lab exercise. OpenMP provides a "relaxed-consistency" and "temporary" view of thread memory. OpenMP specifies nothing about parallel I/O. This is particularly important if multiple threads attempt to write/read from the same file.

IV. SIMULATION RESULTS

ISSN: 2278 – 909X

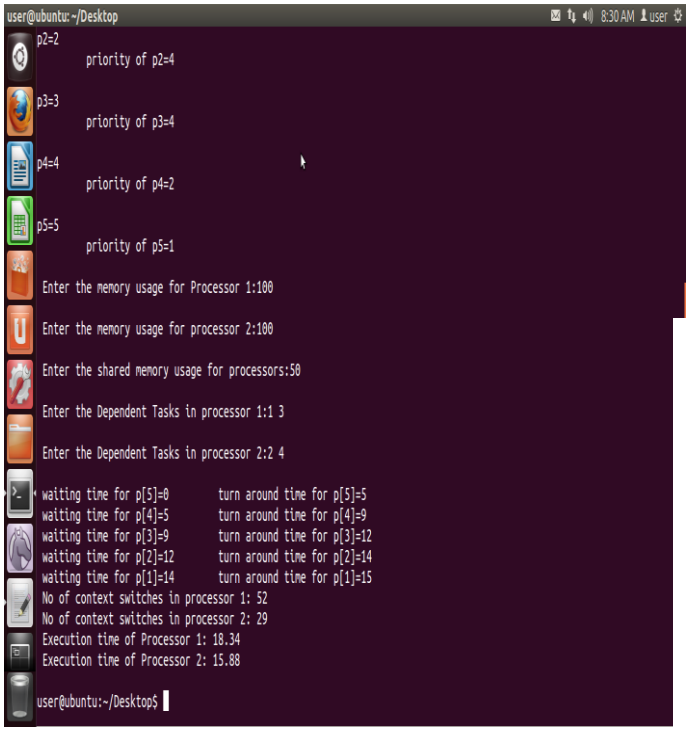

Fig 3: Simulation Output

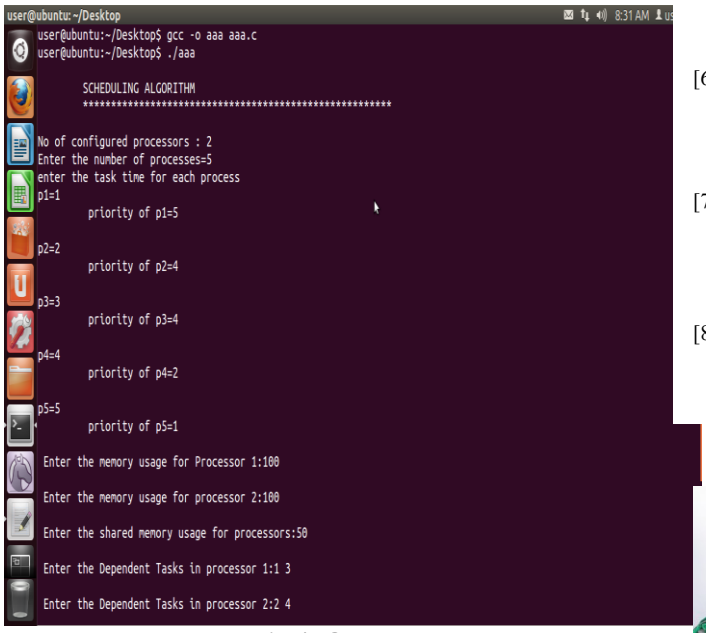

Fig 4: Output

V. CONCLUSION

The design method is to allocate CPU resources in a multicore processor system with the goal of lowering the global power budget and creating a minimal performance loss. The power consumption and performance loss is reduced by using the rate monotonic algorithm. It having static priorities assigned according to the rate monotonic conventions. In future the hardware module has to be implemented. To run three tasks simultaneously and the output will be obtain.

ACKNOWLEDGMENT

First of all we sincerely thank the almighty who is most beneficent and merciful for giving us knowledge and courage to complete the project work successfully. We also express our gratitude to all the teaching and non-teaching staff of the college especially to our department for their encouragement and help done during our work. Finally, we appreciate the patience and solid support of our parents and enthusiastic friends for their encouragement and moral support for this effort.

REFERENCE

[1] OpenMP. OpenMP specification for parallel programming. http://openmp.org.1.3.1, 2.3, 2.5

[2] http : //en.wikipedia.org/wiki/Multicore processor.

[3] G.M. Almeida, and F.G. Moraes, ‘‘Predictive Dynamic Frequency Scaling for Multi-Processor Systems-on-Chip,’’ in Proc. IEEE ISCAS, May 2011, pp. 1500-1503.

[4] V. Chaturvedi and G. Quan. Leakage conscious vs scheduling for peak temperature minimization. In Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific, pages 135 –140, jan. 2011.

[5] A. Fedorova, D. Vengerov, and D. Doucette, ‘‘Operating System Scheduling on Heterogeneous Core Systems,’’ in Proc. Oper. Syst. Support Heterogeneous Multicore Architect., 2007, pp. 1-9.

[6] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," presented at Parallel and Distributed Processing Symposium, Nice, April 22-26, 1003.

[7] N. Guanyz and M. Stiggey, "Fixed-priority multiprocessor scheduling with liu \ layland’s utilization bound," in *Proc. Real-Time and Embedded Technology and Applications Symposium*, Stockholm, 2010, pp. 165 - 174.

[8] Li, J. and Martinez, J. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. The 12th International Symposium on High Performance Computer Architecture (Austin, Texas, February 2006).

Pursuing M.E (Embedded System Technologies) from Sri Shakthi Institute of Engineering and Technology and received B.E. (Electronics and Communication) from Chettinad College of Engineering and Technology.

received M.E. in Applied Electronics from Anna university. Currently working as an assistant professor in Sri Shakthi Institute of Engineering and Technology.

ISSN: 2278 – 909X