Design and Development of Real Time Scheduler Simulator with Co-processor (Teaching Tool)

R. Ramesh and Yaashuwanth Calpakkam

Abstract -- The main objective of this paper is to develop a model for the simulation of real time tasks. The proposed model can be used for preprogrammed scheduling policies. This model provides user friendly Graphical User Interface (GUI). Task ID, deadline, priority, period, computation time and phase are the input task attributes to the scheduler simulator and chronograph imitating the real-time execution of the input task set and computational statistics of the schedule are the output. The proposed framework for the scheduler simulator which is mainly developed can be used as a teaching tool. Evaluation of the performance of real-time schedulers can be done as well. Evaluation criteria is based on the mean response time, number of deadline misses, processor utilization, number of preemptions and context switches. The Task Scheduler Co-Processor is intended to be a task controller. It will perform the controlling function in accordance with a specified scheduling policy. Within the co-processor, a number of scheduling policies may be made available. Scheduling and task switching decisions are made by, and within, the unit, using a policy selected by the application programmer.

Index Terms— Real Time Scheduler Simulator, Task, GUI , Co-Processor , Teaching tool

I. INTRODUCTION

The scheduler is the core part of the operating systems, which orders the assignment of CPU and the resources to the tasks in a multitasking environment. The function of the scheduling algorithm is to determine, for a given task set, a sequence of task step executions (a schedule). The tasks can be classified according to their arrival style: periodic tasks and aperiodic tasks. The task set can be composed of a set of independent tasks if their executions are not synchronized, or the task can be composed of a set of dependent tasks, if their executions are synchronized. If a task set can be scheduled to meet given pre-conditions, the task set is termed as a feasible one. A typical pre-condition for hard real-time periodic processes is that they must always meet their deadlines. An optimal scheduler is able to produce a feasible schedule for all feasible task sets confirming to a given pre-condition. For a particular task set an optimal schedule is the best possible schedule according to some pre-defined criteria.

Manuscript received April, 2011, revised August 10, 2011. This work was supported in part of the project funded by Council of Scientific & Industrial Research (CSIR), Govt. of India.
Dr.R.Ramesh is with the Department of Electrical and Electronics Engineering, Anna University Chennai, Chennai 600 025. (e-mail: rramesh@annauniv.edu).
C. Yaashuwanth is Research scholar with the Department of Electrical and Electronics Engineering, Anna University Chennai, Chennai 600 025. (e-mail: yaash_success@yahoo.co.in).

The purpose of task scheduling is to organize the set of tasks ready for execution by the processor system to meet the performance objectives[3]. The order of these tasks is called a 'schedule'[1]. For real-time embedded systems, the primary objective is to ensure that all tasks meet their deadlines. A schedule can be feasible or optimal: a feasible schedule orders tasks making them to meet all their deadlines; an optimal schedule is one which ensures that failures to meet task deadlines are minimized. The scheduler is responsible for coordinating the execution of several tasks on a processor. The scheduler may be preemptive or non-preemptive. The scheduler for hard real-time systems must coordinate resources to meet the timing constraints of the physical system which implies that the scheduler must be able to predict the execution behavior of all tasks within the system[4]. So the basic requirement of real-time systems is predictability. Unless the behavior of a real-time system is predictable, the scheduler cannot guarantees that the computation deadlines of the system will be met.

Real-time simulation tools speed up the decision making processes during the selection of suitable scheduling algorithm for a real-time embedded application. They also stand as teaching tool helping learners grasp the core ideas related to system modeling quickly There have been various simulator frameworks created for this purpose, [2],[5],[7]. The performance analyses of the simulator frameworks, carried out the need for developing a new Web-based simulator framework which was discovered. The study of existing frameworks of simulation clearly reveals that each tool is better in its own way. Thus an appreciable combination of values of tool is chosen and an earnest attempt has been made to make the proposed framework to be more flexible for the future users for trying different other combinations of evaluation criteria that may be of interest for different real-time resource capacities. The experimentation results obtained from the analyzed simulators are compared to form the reference data for functional verification of the tool under development.

These offers a way for programmers to predict, in advance, whether a multi-tasking design will meet its deadlines or not. Early work is limited to 'rate monotonic' task priorities with deadlines equal to periods, and used a notion of 'processor utilization' to assess schedulability. More recent works have extended schedulability analysis to apply to any fixed-priority scheduling policy, and to support arbitrary deadlines. These works test is for schedulability by calculating the worst-case response time for each task. Sergio Saez et al, [8] faced the overhead problem by introducing a complete hardware architecture that implements slack stealing in hardware using an optimal

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

algorithm that has been completely redesigned to perform efficiently in hardware.The proposed solution which is described by the author is a circuit that behaves as a kind of sophisticated interrupt controller that takes the task workload and the interrupts as inputs, and provides an output to inform the CPU which is the highest priority task. The design of the scheduler hardware was studied from the work of J.E.Cooling and P.Tweedale[6]

A new Real-Time Scheduler Simulator and a Task Scheduler Co-Processor are proposed in this paper which can extensively be used as a Teaching Tool. The objectives of the work are:

- To simulate the scheduling of real-time tasks .
- To execute various scheduling algorithms for tasks.
- To study the performance of various traditional real-time scheduling policies and to submit a performance analysis report to ease the selection of real-time scheduler for a given system.
- o create a graphical user interface based on real-time scheduling simulation framework.
- To develop a hardware setup to practically demonstrate the scheduling theory pertaining to operating systems and Real-Time Operating Systems.

II. PROPOSED ARCHITECTURE

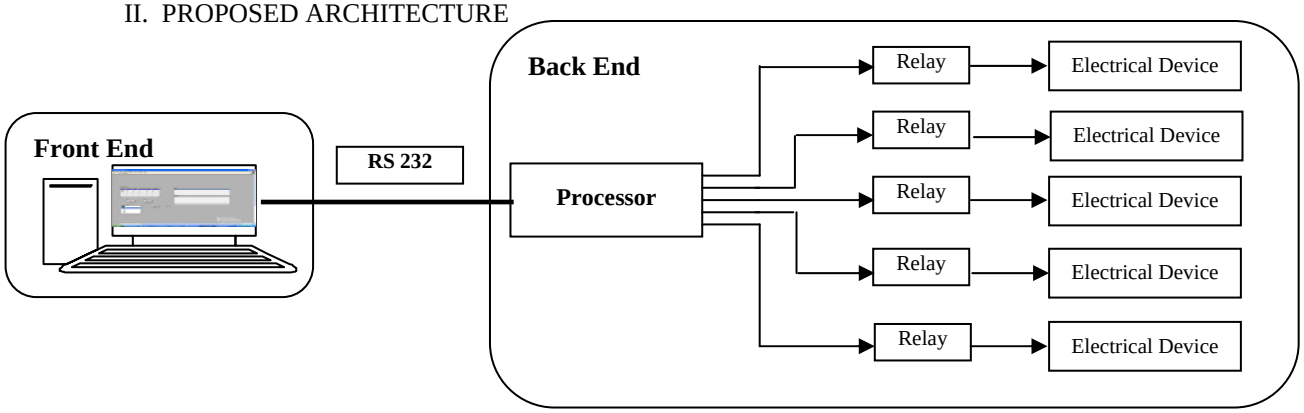

Fig. 1 Block diagram of the simulator The block diagram of the simulator with its hardware setup is shown in figure 1. The simulator consist of front end and back end

The front end of the simulator is GUI based and acts as user interface. The software in the front end of the system gets the value from the user and transfers the value to the processor via serial port. The processor gets the values from the system performs the requested function in the hardware and displays the output in the form of chronographic graph to the user interface.

FRONT END SOFTWARE DESIGN

This section describes the development of the proposed simulator framework in LabVIEW. Real- Time Scheduler Simulator is being developed to be used for teaching real-time scheduling as well as to test and evaluate the real-time scheduling policies used in embedded real-time applications A framework for evaluation of a scheduling algorithm must satisfy characteristics such as simplicity, compatibility with the PC platform and the used operating system, usage of the standard operating system functions, accuracy of results, ease of use, etc. Majority of these requests are aimed for use in the visual user interface that looks as shown in the Figure 2

Scheduling algorithm evaluation and analysis tool performs the task definition, task sets generation, execution of selected algorithms, execution, analysis of the execution and results displaying. The performance evaluation of the real-time scheduling algorithms is carried out based on the results obtained through computational analysis. Various stages of evaluation procedure are: Identification of the tasks, Task set implementation, Selection of algorithms, Simulation Timing definition, Simulation execution and Scheduling algorithms evaluation

The front end of the simulator is GUI based and acts as user interface. The software in the front end of the system gets the value from the user and transfers the value to the processor via serial port. The processor gets the values from the system performs the requested function in the hardware and displays the output in the form of chronographic graph to the user interface.

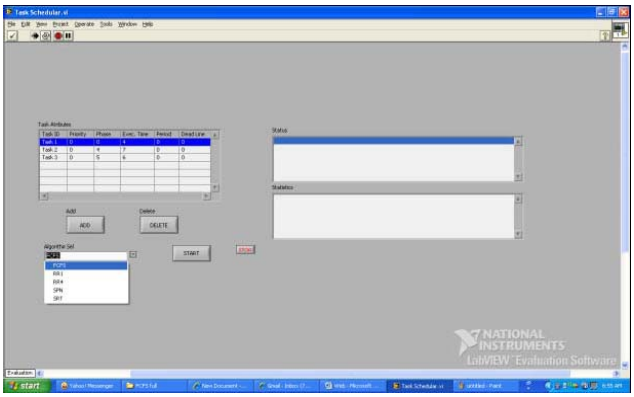

Fig.2 Software front end of the simulator

The most successful scheduling algorithm for the periodic tasks scheduling is the one that has minimal response times, minimal number of tasks with missed deadlines and maximal resource utilization in the given workload and with other parameters. The complete task

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

model is too complex for implementation and some of the task parameters are hence ignored. In real-time systems two characteristics of tasks are considered to be of primary interest: criticality or importance; and timing. Task importance is frequently a subjective issue, whereas timing is objective. The essential timing attributes of tasks are deadline (TD), worst-case computation time (Tcw) and period (Tp),

Elements of the simulator Task Attributes are part of the simulator that allows the user to add Task (to the existing task set) with parameters like Task ID, Priority, Phase, Execution Time, Period, and Deadline. Among these parameters, Task ID should be unique for each input task. All other parameters are numerals. All the parameters are required to save a new task to a task set, modify the parameters of a task from the declared task set. delete task(s) from the input task set Fig.5.1 shows the elements of simulator.

1) Resource Usage: Add a resource to specific task(s) for its execution (critical section) with Resource ID, Task ID, Start, and Time. Fig.3 shows the adding and deleting of tasks

2) Simulator Controls: The Simulator Controls part of the simulator is the main part of the simulator which provides the following functions to the user: Selection of Scheduling Policy, selection of Task Synchronization Protocol, choice of preemptive or non-preemptive scheduling and run Feasibility Test to find whether the declared task set is feasibly schedulable with the desired scheduling scheme The software output is shown in figure 3

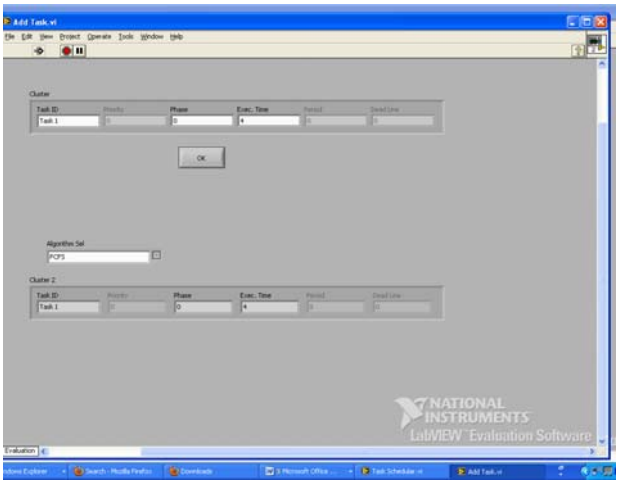

Fig.3 Adding and deleting tasks in a simulator

BACK END HARDWARE DESIGN

Hardware Design for the simulator inputs are got from the front end through serial port connection the target processor receives the values from the front end of the system. This target processor controls all the electrical devices on receiving the input performs the commands and also displays the result in the front end of the system for user to view it. LCD display connected in the hardware denotes which task is currently being executed by the processor

Hardware software integration

A special proprietary software called win x Talk 1.5 can be used to set the baud rate of serial port and also used to see the execution of tasks by the processor Figure 4 shows Win x talk 1.5 showing the execution of tasks by the processor

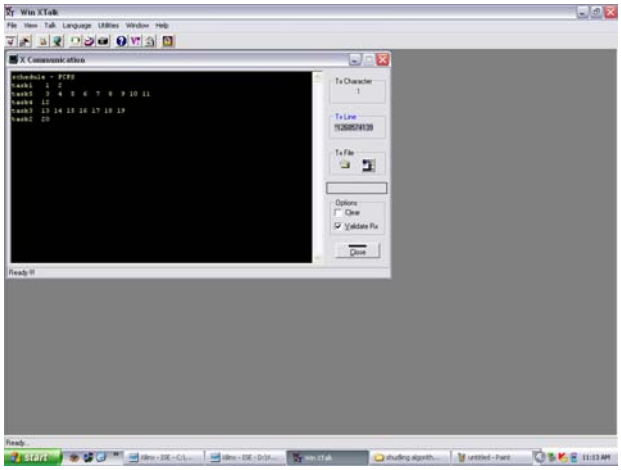

Fig. 4 Execution of tasks in win x talk 5 software

Sample output

The processor executes the required task as the input is given from the front end. A lcd display which is connected to the hardware shows which task is currently executed by the processor. A chronographic graph and also the status of execution of task by the processor is also shown in the front end in figure 5 and figure 6 shows the output of simulator in both hardware and software

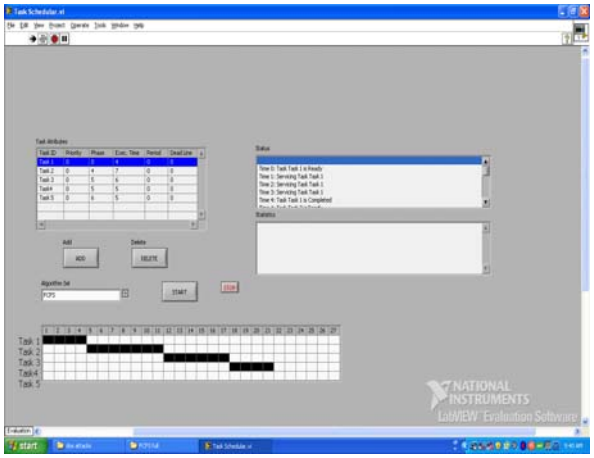

Fig. 5 Output view – Software

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

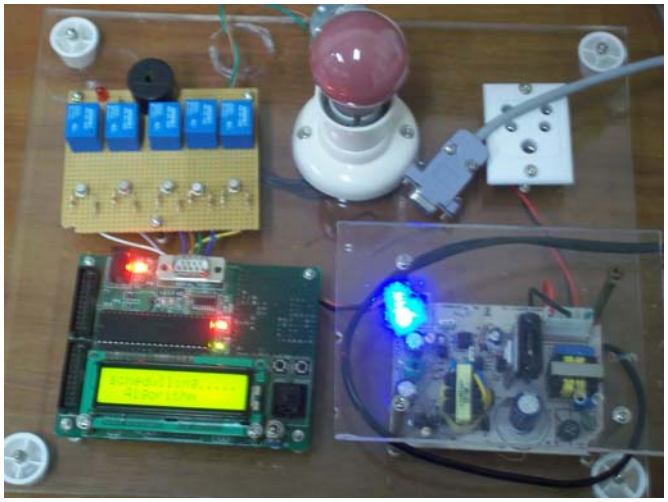

Fig.6 Output view – Hardware

III. CONCLUSION

The framework which is proposed can be used as an invaluable teaching tool. Further, the GUI of the framework will allow for easy comparison of the framework of existing scheduling policies and also simulate the behavior and verify the suitability of custom defined schedulers for real-time applications. In addition, the real-time scheduler co-processor hardware can help the learners have a closer view of scheduling tasks in real-time hardware. Future work includes implementation of multiprocessor and aperiodic real-time schedulers in the simulator for exploring the full spectrum of real-time scheduling theory

REFERENCES

[1] A.Burns "Fixed priority scheduling with deadline prior to completion" Real-Time systems Research Group Department of computer science university of york,.pp138-142 UK June 1994.

[2] Arnoldo Diaz, Ruben Batista and Oskardie Castro, Realtss: "A real time scheduling simulator," in 4th International Conference on Electrical and Electronics Engineering (ICEEE), , pp. 165-168, Sep.2007 Mexico City, Mexico.

[3] Barbara Korousic-Seljak, "Task scheduling policies for real-time systems, in Microprocessors and Microsystems," *Vol.18 No. 9*. pp.501-511, Nov.1994.

[4] Chi-scheng shih, Lui Sha and Jane Liu, "Scheduling Tasks with Variable Deadlines," Proceedings Seventh IEEE. Real-Time Technology and Applications Symposium, 2001. IEEE, pp. 120-122 June 2001 Taipei.

[5] F. Singhoff, J. Legrand, L. Nana, L. Marce, Cheddar: "A Flexible Real Time Scheduling Framework," in ACM SIGADA'2004 International conference proceedings, Atlanta, Giorgia, USA. Nov. 2004.

[6] J.E. Cooling, P. Tweedale, Task scheduler co-processor for hard real-time systems, in Microprocessors and Microsystems, *Vol. 20 No 9* , pp. 553-566, Dec 1996.

[7] Jan Blumenthal, Frank Golatowski, Jens Hildebrandt, Dirk Timmermann, "Framework for Validation, Test and Analysis of Real-Time Scheduling Algorithms and Scheduler Implementations," in Proceedings of the 13th IEEE International workshop on Rapid System Prototyping (RSP'02), 2002.

[8] Sergio Saez, Joan Vila, Alfons Crespo and Angel Garcia "A Hardware Scheduler for Complex Real-Time Systems". Proceedings of the IEEE International Symposium on Industrial Electronics IEEE Nov 1999 pp 43 -48.

ACKNOWLEDGEMENT

The authors would like to acknowledge financial support of Council of Scientific & Industrial Research (CSIR), Govt. of India.

ABOUT THE AUTHORS

Mr. C.Yaashuwanth completed M.E. in Embedded System Technologies at Anna University Chennai. He is currently Pursuing his Ph.D program under Dr. R .Ramesh in Department of Electrical and Electronics Engineering College of Engineering Guindy Anna university Chennai.

Dr. R. Ramesh is currently Assistant Professor in Department of Electrical and Electronics Engineering College of Engineering Guindy Anna university.He received his Ph.D. degree from Department of Electrical and Electronics Engineering College of Engineering, Guindy Anna University, Chennai. His area of interest are Real-time Distributed Embedded Control On-line power system analysis and Web services.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)