Teaching Real Time System Scheduling using low cost microprocessor board

Subramaniam Ganesan[1], Xuewen Ding[1], **Andrew Rusek**[1]
[1] Department of Electrical and Computer Engineering
Oakland University, Rochester, MI 48309

Emails: ganesan@oakland.edu, rusek@oakland.edu, , dingxw1@126.com

Abstract

Objectives of the Real-Time Systems course reflect what we expect the students to learn from the course. We make sure that it is possible to assess the objectives at the end by internal or external expert. We need to make sure that the teaching activities and feedback from the students are synchronized with the objectives.
One of the objectives of this course is: ´Know the various task assignment and scheduling methods. For example RM and EDF scheduling"

We describe here a simulation software tool and a low cost ARM microprocessor board with MDK-ARM Keil development tool to teach the scheduling concepts effectively. A number of experiments to learn real-time task scheduling concepts are explained. Use of Real-Time scheduling in automotive embedded system applications is described and justifies the importance of our teaching techniques. This emphasizes active learning instead of passive learning.

Introduction

Real time system course is a graduate level course and it emphasizes hard and soft real time computer system design for uniprocessor embedded system applications and distributed real time systems [1-6]. Topics covered include characterizing real-time systems, performance measure, task assigning, scheduling, Fault tolerant scheduling, run-time error handling, run-time support, kernel, real time databases, real-time communication, software development techniques; practical applications.

Course Objectives are:

Upon completion of this course students will be able to:

1. Know the definition and characteristics of Real Time systems
2. Know the various task assignment and scheduling methods. For example RM and EDF scheduling.
3. Become familiar with Real Time system development tools like Matlab RT tool box, ETAS tools
4. Know the important characteristics of Real Time Operating System
5. Know about the RT System requirement, design, and performance analysis.

In the following sections, we show how we make sure that the students learn the objective number two on scheduling. Details of course topics are given in the appendix A.

Scheduling

There are many task scheduling algorithms and each of them makes sure that the tasks are schedulable and they meet the expected deadline [8-11]. Three most popular algorithms are defined below.

Rate Monotonic Scheduling Algorithm

The rate-monotonic scheduling algorithm schedules periodic tasks using a static priority policy with preemption. If a lower-priority process is running and a higher-priority process becomes available to run, it will preempt the lower-priority process. For each process, its priority=1/pi(inverse of the period).

Deadline Monotonic Scheduling Algorithm

Deadline Monotonic Scheduling Algorithm schedules periodic tasks or aperiodic tasks using a static priority policy with preemption. For each process, its fixed priority = 1/di(inverse of the relative deadline). That is, the shorter the relative deadline, the higher the priority.

Earliest-Deadline-First Scheduling algorithm

Earliest-Deadline-First scheduling dynamically assigns priorities according to deadline. The earliest the deadline, the higher the priority. Algorithm: At each moment of time t, schedule the task whose deadline is closest to t. Ties can be resolved arbitrarily

Scheduling Simulators

We describe below task scheduling simulators that are free and used for teaching scheduling. One is called CPUSS [12] and the other is Cheddar [13].

CPU Scheduling Simulator (CPUSS) is a framework that allows you to quickly and easily design and gather metrics for custom CPU scheduling strategies. CPUSS records the following metrics about your scheduling algorithm:

- Average process wait times
- Idle CPU time
- Busy CPU time
- Wait time mean
- Wait time standard deviation
- Response time mean

- Response time standard deviation
- Turnaround time mean
- Turnaround time standard deviation
- Throughput stats
- Throughput mean
- For each process
    - Arrival time
    - Start time
    - Completion time
    - CPU activity
    - Burst time
    - Id
    - Priority
    - Wait time
    - Turnaround time
    - Response time

The core features CPUSS allows to:

- Define the processes to schedule
- Auto generate the processes to schedule (varying burst time properties)
- Log results to SQL Server
- Hook into events
    - Simulation Started/Completed
    - Process Started/Preempted/Resumed/Completed

CPUSS support the following strategies:

- First Come First Served
- Round Robin (time quantum can be defined)
- Shortest Job First*
- Priority First*
- SJF with Priority Elevation rule (threshold can be defined)*

CPUSSRG allows to quickly and effeciently generate a report and view some of the key stats from the simulation (avg/std dev/var wait times, cpu utilization etc). One can run the simulation many times to get a broader picture of the data for further analysis.

Cheddar Real Time Simulator [13].

Cheddar is a free real-time scheduling tool. Cheddar allows you to model software architectures of real-time systems and to check its schedulability or others performance criteria. With the schedulability analysis tool, schedulability can be accessed with both scheduling simulation and with feasibility tests. Also new real-time scheduling policies or task models can be analyzed. Cheddar is composed of two independent parts : an editor used to model the real-time system to

analyze, and a framework to perform such analysis.

The editor allows to describe systems composed of several cores, processors which own tasks, shared resources, buffers and which may exchange messages or communication with buffers. Cheddar includes its own ADL, namely Cheddar ADL. The framework includes many feasibility tests and simulation tools. The main analysis tools of Cheddar are the following :

- Analysis with scheduling simulations :
  - With preemptive and non preemptive scheduling policies
  - With uniprocessor and multiprocessor scheduling policies
  - With uniprocessor : Rate Monotonic, Deadline Monotonic, Least Laxity First, Earliest Deadline First, POSIX queueing policies (SCHED_OTHERS, SCHED_FIFO and SCHED_RR), Maximum Urgency First, Round-Robin and several time sharing scheduling policies.
  - With multiprocessor : global version of uniprocessor scheduling policies, Proportionate Fair, EDZL, RUN
  - With instruction cache entities
  - With different type of tasks : aperiodic, periodic, task activated with a Poisson process, ...
  - With shared resources (and with FIFO, PCP, PIP, IPCP synchronization protocols)
  - Several hierarchical schedulers, such as ARINC 653 scheduling, sporadic server, polling server, deferrable server
  - With task jitters and offsets
  - With various task precedencies
- Extract information from scheduling simulation, such as :
  - Worst/best/average task response times, task missed deadlines
  - Number of preemption, number of context switch
  - Worst/best/average shared resource blocking time
  - Deadlock and priority inversion
  - Worst/average buffer utilization factor, message worst/average waiting time
- Apply feasibility tests on tasks, buffers and shares resources :
  - Compute worst case task response time on periodic task set
  - Several methods to compute worst case response time with linear and tree transaction
  - Apply processor utilization feasibility tests.
  - Compute bound on buffer size (when buffers are shared by periodic tasks)
  - Worst case shared resource blocking time
  - Memory footprint of software entities
- Shared resources support (both scheduling simulation and blocking time analysis). Supported protocols : PIP, OPCP, IPCP
- Tools to express and do performance analysis with task dependencies :
  - Model task transaction (linear or tree) and compute worst case response time
  - Scheduling simulation tasks according to task precedencies
  - Compute Tindell Holistic end to end response time.
  - Apply Chetto and Blazewicz algorithms.

- Task and resource priority assignment :
    - Classical Rate Monotonic, Deadline Monotonic, Audsley task priority assignment
    - Task priority assignment according to CRPD
    - Shared resource ceiling priority assignment (for PCP like policies)
- Features to allow users to define and handle their own policies :
    - User-defined scheduling policies (based on pipeline models or automaton models)
    - User-defined task model policies, to express and handle specific task activation policies
    - User-defined analysis on scheduling simulation, to look for specific properties on scheduling simulation result
- Partitioning algorithms for periodic task set :
    - Best fit policy
    - General Task fit policy
    - Fist fit policy
    - Small fit policy
    - Next fit policy

ARM Microcontroller board with RTOS for Task Scheduling experiments

STMicroelectronics has many STM32 family of 32-bit Flash microcontrollers and application development boards at low cost 14,15]. The new boards are fully compatible with the existing STM32 development ecosystem, including the range of dedicated plug-in application expansion boards that allow specialized features ranging from motor drives to motion and environmental sensors to be easily incorporated into the final application. Moreover, they offer unlimited extension capability via three types of connector: in addition to the Arduino™ Uno and ST morpho connectors provided by existing Nucleo-64 boards, the new boards include an ST zio connector. Together, these three connectors give complete access to all of the STM32 general-purpose I/O pins, allowing easy implementation of any creative function. Selected STM32 Nucleo-144 boards include Ethernet, as well as USB FS OTG ports to ease connections to local/wide area networks.

The STM32 Nucleo-144 boards allow the entire embedded processing community, from hobbyists and students to the most experienced professional system developers, to rapidly test, optimize, and industrialize new applications built around the industry's most popular, affordable, and efficient 32-bit microcontroller family. They are compatible with the most popular development toolchains including IAR EWARM, Keil MDK-ARM, and GCC/ LLVM-based IDEs such as AC6 SW4STM32 or Atollic TrueStudio.

The first four boards (NUCLEO-F746ZG, NUCLEO-F429ZI, NUCLEO-F446ZE and NUCLEO-F303ZE) are available now at US$23 for the Ethernet versions and US$19 without Ethernet

Keil MDK Version 5 is the latest release complete software development environment for a wide range of ARM, Cortex-M, and Cortex-R based microcontroller devices [16,17]. MDK includes the µVision IDE/Debugger, ARM C/C++ Compiler, and essential middleware components. It's easy to learn and use. Keil RTX deterministic, small footprint real-time operating system (with source code), RTOS and supports a few task scheduler, Execution Profiler and Performance Analyzer. Numerous example projects help you quickly become familiar with MDK-ARM's powerful, built-in features

Task Scheduling in Automotive

Today's cars have nearly 40 to 100 microcontrollers to control various functions (Example, Engine controller, Transmission controller, Display controller, Anti-lock brakes, etc.) [18]. Each controller collects data from various sensors at certain interval, process them, uses the results to control outputs and also communicates the sensor values over the CAN bus to other controllers. This shows that each controller have many tasks to complete at various interval before the desired deadlines. The students choose some of these controllers and choose multiple task's arrival time, execution time, dead line and try to simulate using the simulators and make sure that the tasks meet the deadlines. They also try to test their concepts using the low cost microcontroller boards.

Student Participation in Experiments

The students attending the real time systems course do a number of simulation exercises and also develop small projects using low cost microcontroller boards (example: Traffic controller, bottling plant controller etc. These activities help them learn the task scheduling, feasibility analysis, and implementation. This emphasizes active learning instead of passive learning and also increases creative thinking and productive thinking [19, 20]. They also do oral presentation on their project and submit a project report.

Conclusions and how the Students Benefit from this Approach

The lab exercises and projects introduced in the course and simulation exercises have made the course material easy to understand and fun to learn the application of various scheduling algorithms. The course evaluations on the course outcomes received very good results.

The concepts learned in this course are useful for graduate students to use in the automotive embedded system design and testing. This paper provided an overview of simple and effective ways to teach task scheduling algorithms for real time applications.

References:

1. Phillip A. Laplante, "Real time systems design and analysis", 4th edition, Wiley InerScience, ISBN 978-0-470-76864-8
2. Jane W. Liu "Real Time Systems" Prentice Hall, 2000, ISBN: 0-13-099651-3.
3. C.M. Krishna and R.G. Shin, "Real time system" McGraw Hill 1997.
4. Micro C/OS-II, The real time kernel, A complete portable, ROMable, scalable preemptive RTOS by Jean J. Labrosse, R&D books, Miller Freeman inc., ISBN: 0-87930-543-6; Phone: 785 841 1631.
5. Embeded Systems Building Blocks, 2nd edition, Complete and ready to use modules in C, by Jean J. Labrosse, R&D books, Miller Freeman inc., ISBN: 0-87930-604-1; Phone: 1-800-788-3123.
6. Jeffrey Tsai and Steve Yang, " Monitoring and Debugging of Distributed real time system" IEEE computer Soceity press, ISBN 0-8186-6537-8
7. Java for Embedded System, by Ingo Cyliax, Circuit Cellar magazine, December 2000 and January 2001.
8. Real Time JVM, New Monics Inc., www.newmonics.com
9. Jworks, Windriversystems, Inc, www.wrs.com
10. Java Chip, ajile systems inc., www.ajile.com
11. Valvano, " Embedded microcontroller system- real time interfacing" Brooks/Cole publisher
12. https://cpuss.codeplex.com/
13. http://beru.univ-brest.fr/~singhoff/cheddar/
14. http://www.st.com/web/en/news/n3784
15. http://www.keil.com/arm/mdk.asp
16. http://www.arm.com/support/university/
17. Embedded Systems: Real-Time Interfacing to Arm® Cortex(TM)-M Microcontrollers 2nd Edition by Jonathan W. Valvano , July 2015, http://users.ece.utexas.edu/~valvano
18. Ronald Jurgen " Automotive Handbook", McGrawHill Handbook, second edition.
19. http://ericbrown.com/critical-thinking-vs-creative-thinking.htm
20. Think Better: An Innovator's Guide to Productive Thinking by Tim Hurson, McGraw Hills companies, ISBN-13: 978-0-07-149493-9

Appendix A: Real Time Systems Course details

Text book: Phillip A. Laplante, "Real time systems design and analysis", 4th edition,
Wiley InerScience, ISBN 978-0-470-76864-8
Reference books:
Jane W. Liu "Real Time Systems" Prentice Hall, 2000, ISBN: 0-13-099651-3.

C.M. Krishna and R.G. Shin, “Real time system” McGraw Hill 1997.

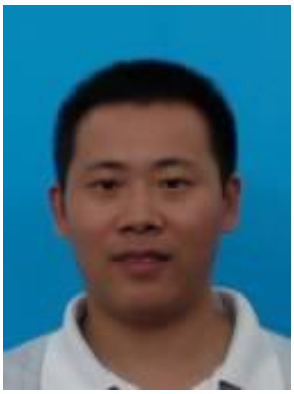
COURSE Topics:

1) Introduction and Basic concepts (1 hr)
2) Characterization of Real time systems and Tasks (2hr)
   Reference model of Real Time Systems
3) Task Assignments and Scheduling (6 hr)
   Real time scheduling
   Clock-driven scheduling
   Priority driven Scheduling
   Multiprocessor Scheduling
4) Real Time Tools
   Real time Java, Matlab Real Time Tool Box , RTAI from ETAS (1hr)
5) Real time Operating Systems
   RT kernels, Intertask communication and Synchronization
   Memory Management
   Case Study: Win CE, Real time Linux, QNX
   UC/OS II real time kernal (8 hrs)
6) Real time Communications (4 hrs)
7) Distributed real time systems (4 hours)
8) Fault Tolerance (1hr)
9) Real time DSP System
   Code composer Studio
   Development Platform (6 hrs)
10) Real Time System Development, Code Warrior software (1 hr)
11) Performance Analysis and optimization (3 hrs)
12) Divisible Load Theory
11) Automotive Applications (4 h)

Biography

Subra Ganesan (ganesan@oakland.edu) is a Professor of Electrical and Computer Engineering at Oakland University and Director of Real Time Embedded DSP Systems Lab. He joined the university in 1984. After graduating from Indian Institute of Sciences Bangalore India, he served at universities in Germany, and Canada and Indian research laboratory as a scientist. He does research work in collaboration with TI, Free Scale, and a few automotive companies and US army. His research areas include DSP, Embedded Systems, Real Time Systems, Condition-based Maintenance, and Optimization.

Andrew Rusek (rusek@oakland.edu) is a Professor of Engineering at Oakland University in Rochester, Michigan. He received an M.S. in Electrical Engineering from Warsaw Technical University in 1962, and a PhD. in Electrical Engineering from the same university in 1972. His post-doctoral research involved sampling oscillography, and was completed at Aston University in Birmingham, England, in 1973-74. Dr. Rusek is very actively involved in the automotive industry with research in communication systems, high frequency electronics, and electromagnetic compatibility. He is the recipient of the 1995- 96 Oakland University Teaching Excellence Award



Xuewen Ding (dingxw@tute.edu.cn) is an Associate Professor of Electrical Engineering School at Tianjin University of Technology and Education, and Director of Electrical Engineering Department. He received an M.S. in Signal and Information Processing from Tianjin University in 2003, and a PhD. in Signal and Information Processing from the same university in 2008. He worked as a visiting scholar at Oakland University in 2015. Dr. Ding is very actively involved in Intelligent Information Processing. His research areas include Computer vision, Machine learning and Real Time Systems.