ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization *Volume 6, Special Issue 14, August 2017*

Two Days National Conference on Modernization in Engineering Science (NCMES 2K17)

23rd-24th March 2017

Organized by

Dept. of EEE, Sri Krishna College of Engineering & Technology, Coimbatore 641 008, Tamil Nadu, India

VLSI based memory peripheral for RTOS optimisation

S.Sheeba Rani Gnanamalar M.E., MBA.,(Ph.D)[1], A.Anie Selva Jothi M.E [2]

Assistant Professor, Department of Electrical and Electronic Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, India[1]

Assistant Professor, Department of Electronics and Communication Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, India [2]

ABSTRACT: The growing technology in embedded application has resulted in use of increase in RTOS. Unfortunately RTOS can introduce significant performance degradation due to the following reasons: 1) when we implement real time task in software the processor utilization is more in order to schedule and manage the tasks. Hence lot of processor utilisation is wasted.2) The RTOS occupies considerable memory space to complete the task scheduling.3) Power consumption is more due to read/write operations of the memory and switching activities of the memory cycle.4) Speed of execution is dependent on the processor clock.This paper presents the Real-Time Task Manager (RTM)—a processor extension that minimizes the above performance drawbacks associated with RTOSes. The RTM accomplishes this by supporting, in hardware, a few of the common RTOS operations that are performance bottlenecks: task scheduling, time management, and event management. By exploiting the inherent parallelism of these operations, the RTM completes them in constant time, thereby significantly reducing RTOS overhead. It decreases both the processor time used by the RTOS and the maximum response time by an order of magnitude. This hardware support for RTOS is implemented in VHDL and the simulation and synthesis report are generated.

KEYWORDS: RTOS, Scheduling, Hardware-Software Codesign, Processor Utilisation

I. INTRODUCTION

RTOSes havebecomeextremely important tothe development of real-time systems as reflected by the growing market for RTOSes; halfa billion dollarsin shipmentsof RTOSeswill be soldin 2002 [5]. RTOSesprovide services forbetter hardware abstraction multitasking, task synchronization etc. However, the benefits that RTOSes provide do not come for free. The use of RTOSes has several important effects on various performance parameters of real-timesystems.

Processor Utilization: The fraction ofprocessing power that an application is able to use. In order to provide services like multitasking, pre-emption, and numerous others, the RTOS introduces an overhead which lowers the processing power available to theapplication

ResponseTime: The time it takes for a real-time system to response to external stimuli. This delay is highly dependent on several factors, including whether or not the RTOS is pre-emptive and whetherpollingor interrupts areused to sense the stimulus. The effects of RTOSes on response time vary widely, but, in general, RTOSes increase response time to varying degrees.

Much of the performance degradation caused by RTOSes can be traced to their coreoperations,namely task scheduling,time management, and event management. Analysis of these functions reveals that they are executed frequently, that theyperform inter-related actions, and that these actions exhibit parallelism. That parallelismcannot beexploited by software- based implementations; however, hardware-based solutions can exploit this parallelism and lower the performance degradation. Also, the rapidly dropping cost of logic [7] has made it possible to put custom hardware for enhancing frequently executing operations on embedded processors, without increasingcosts significantly.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 6, Special Issue 14, August 2017

Two Days National Conference on Modernization in Engineering Science (NCMES 2K17)

23rd-24th March 2017

Organized by

Dept. of EEE, Sri Krishna College of Engineering & Technology, Coimbatore 641 008, Tamil Nadu, India

These factorssuggestthat RTOSes would greatly benefit from a custom hardware solution. This is the motivation for the Real-Time Task Manager (RTM), a memory-mapped on-chip peripheral designed to optimize task scheduling, time management,and event management thatiscompatible with a wide range of RTOSes. Measurements have been taken of the performance impact of the RTM on models of realistic real time systems. These measurements show that processor utilizationand maximum responsetime arebothreduced by an order of magnitude.

II. RTOS FUNCTIONS

To better understand that the commonly seen bottlenecks in RTOSes,we presentsome of the detailsregarding their nature— what they are, how they are implemented and how often they are invoked.

Task Scheduling: Process of determining which task should be running at any given time. The most commonly implemented approachto scheduling in commercial RTOSesis based on task priorities [8]. Priority-driven schedulers assign each task a priority and execute the task with the highest prioritythat is ready.

TimeManagement: The RTOS gets its sense of time by aperiodic interrupt generated every clock tick by a hardware timer. The rateofthegeneration of this clock tick is the system clock frequency which is not to be confused with the CPUclock frequency. Typically the system clock frequency is on the orderof kHz while the CPU clock is onthe order of MHz. *Time management* refers to the RTOS's ability to allow tasks to use this mechanism to be scheduled at specific times i.e. havethetasksblockforaspecifictimebeforebecomingready.

Event Management: Most real-time operating systems provide services for communication and synchronization between tasks—known as Interprocess *communication (IPC)*. Examples are semaphores, message queues etc. Event management involveskeeping track ofwhichtasks are blocked i.e. waiting for IPC and which tasks should be released i.e. haveto receive IPC on resourceavailability.

Note that as shown inTable 1, theoverheadof theRTOS due to these operationsis proportional to the product of frequency andcomplexity.Since both of these componentsmayincrease linearly withthenumber of tasks, there may be a quadratic relationship between the number of tasks in the system and the RTOS overhead due tothem.In addition the overhead of task scheduling and time management increases linearly with the system clock frequency.

The longest time thatit takes toperform anycritical sectionof code (including task scheduling, event or time management) adds to the maximum response time in pre-emptive systems. However, in nonpre-emptive systems whereinterrupts are polled, a response task is delayed more by executing workloads than by RTOS operations. This is because workloads execute for longer periods thanRTOS operations andthus disable interruptresponse for longer periods.

III. REAL-TIME TASK MANAGER

The RTM is a hardware module that implements a task database and thus supports the key RTOS operations: task scheduling,time management,and eventmanagement. It isan on-chip peripheral that communicates withthe core via a memory-mapped interface. The RTM is pictured in Figure 1; where each record contains information abouta singletask.The RTM is accessedthrough the global address space and each record can be read from or written toas ifitwere anyother array of structures.

A record is composed of four individually addressable fields. Each record contains a*status field*,containing several bitsthat describe the status of the corresponding record. The*valid bit* is necessary to indicate if that the record isusedby a task. The *delayed bit*indicates that the taskiswaiting forthe amountof clock ticks specified by the*delay field* before beingready-to-run. The event *bit*indicates that the task is pending on the event with the identifier specified by the event *ID field*. The *suspended bit* indicates that thetask hasbeen suspended. If the valid bit isset,but the delayed, event, andsuspended bits are clear, thenthe task is ready-to-run. Finally, the priority *field* indicates the task's priority. Also, the maximum number of records is some fixedconstant, such as 64 or 256. The RTOS can issue instructions toor retrievecertain information from the RTMvia a set of memorymappedregisters. The RTM can be queried for the highestpriorityreadytask and thus supports static-priority scheduling. The RTM does time management bydecrementing the delay fieldsof allrecordsby the numberof clock ticks specified by theRTOS and updating the delay bit if this value reaches zero. Event

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 6, Special Issue 14, August 2017

Two Days National Conference on Modernization in Engineering Science (NCMES 2K17)

23rd-24th March 2017

Organized by

Dept. of EEE, Sri Krishna College of Engineering & Technology, Coimbatore 641 008, Tamil Nadu, India

management is very similar tostatic-priority scheduling. TheRTM queries for the highest priority task that is pending on a given event identifier, instead of querying its data structure for the highest priority readytask.

IV. ARCHITECTURE

A reference architecture is presented Above to illustrate the complexity and scalability of implementing the hardware architecture of theRTM.For a64 record RTMwith entriesas shown in the figure the total number of flip-flops required is2304,whichis small enoughto easily be implemented. These parameters are based on common limits and are sufficient for the majority of real-time applications.

Task scheduling is performed using 64Ready Testcells and a binary tree of 63 Priority Test cells. Owing to space considerations we show this for a 32 entry caseinFigure 2. TheReady Test cellssimply determine which tasks areready- to-run. Each Priority Test cell takes in as input twotaskpriorities and ready bits. It then outputswhich ofthese the highest priority ready task is. These priority values propagatedownthecomparatortreeuntilthesinglesixthordercellwhere the index of the overall highest priority ready task is output. Each N^{th} order Priority Test cellcontains an 8-bitcomparator, an8-bit2:1 MUX,an(N-1)-bit 2:1MUX, an OR gate, an AND gate, and a NAND gate. The logic required task scheduling scales linearly with the numberofrecords intheRTM (n RTM records requires n RT cells plus n-1 PT cells). The computational delay is proportional tothe logarithmof the number of records. For a reasonablenumber of recordsi.e. the numbers typically used in production RTOSes, this implementation of task scheduling will be sufficientlyfast and small.

In order to implement time management, 64 delay decrement cells are all that is required, as shown in Figure 3. Delay decrement cells consists of a simplified 16-bit adder and an ANDgate. Delay cells decrementthe delay field and perform a comparison to clear the delay bit when necessary. In this implementation, the logic scales linearly with the number of records; however, the computational delay is a constant. In designs where the CPU clock frequency is lower than the system clock frequency, the delay decrement cells can be reused and thus lowerthenumber ofdelay cellsrequired.
The event management is almost exactly the same as task scheduling. The only difference is that instead of Ready Test cells, it has Event Test cells, as seen in Figure 2. This allows the binary tree of Priority Test cells to be used for both scheduling and event management. The Event Test cells just check if the event bit is set and if the eventIDof the resource being released matches theeventID ina specificrecord.They each use an 8-bit comparator and an AND gate. The combinational logic required for event management scales linearly, except that the majority can be shared with the task scheduler. Also, the computational delay is O (log (n)), as it is with scheduling the die area that theRTM needs is importantindetermining its feasibility.Basedonexisting area models forregister files and caches [9], the RTM reference architecture requires approximately 2600 register-bit equivalents (RBEs). This is roughly equivalent to the die area used by a 32-bitby 64-word register file making it feasible to implement the RTM in hardware. This number islarge compared to a simplecore but issmall comparedtothe whole chip. This isbecausea typical embedded processor consistsof the core and other on-chip devices like memory and various I/O devices.

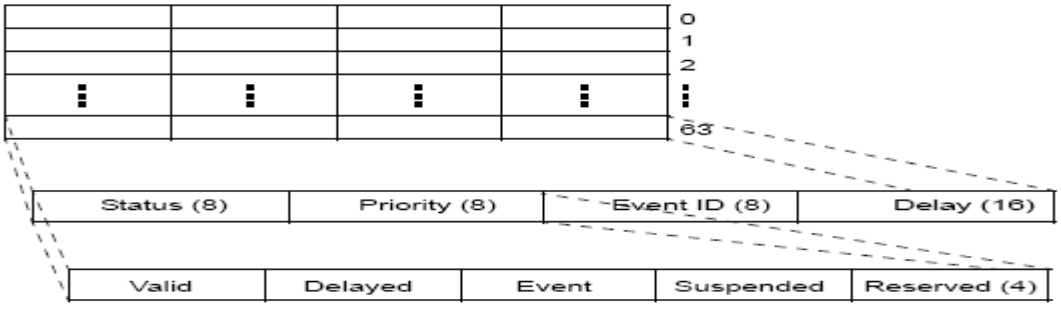

Fig. 1ReferenceRTM Data Structure

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization *Volume 6, Special Issue 14, August 2017*

Two Days National Conference on Modernization in Engineering Science (NCMES 2K17)

23rd-24th March 2017

Organized by

Dept. of EEE, Sri Krishna College of Engineering & Technology, Coimbatore 641 008, Tamil Nadu, India

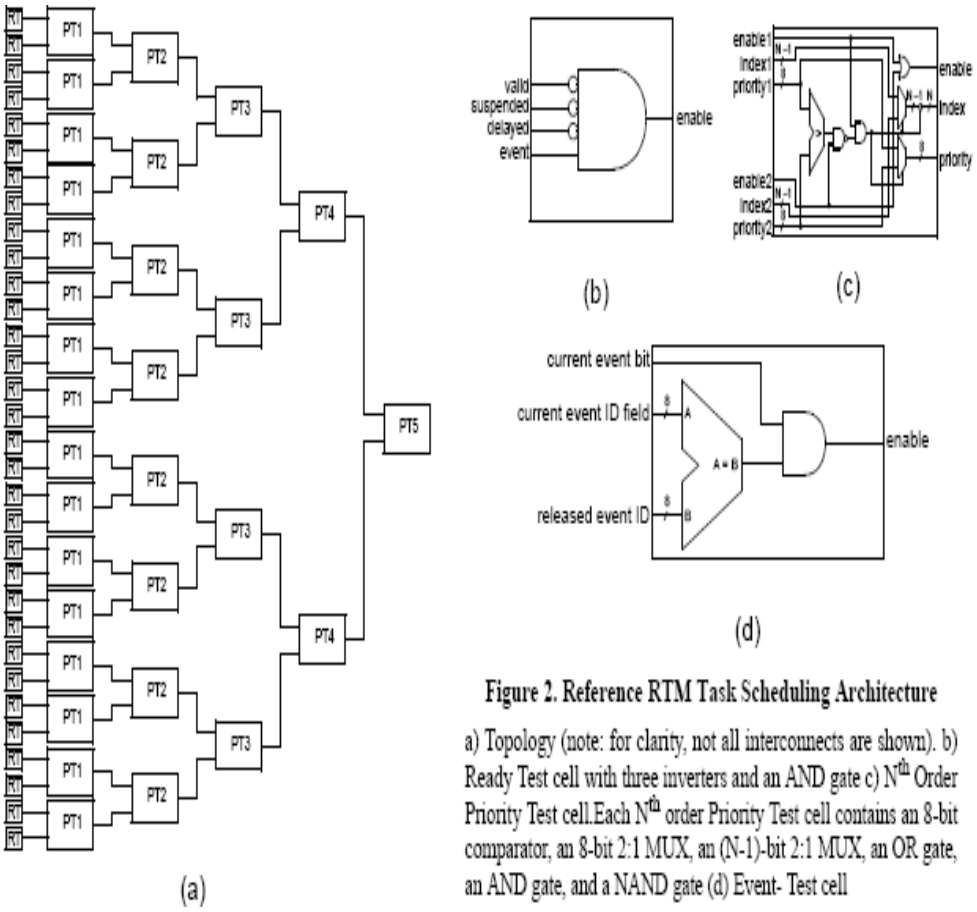

Figure 2. Reference RTM Task Scheduling Architecture

a) Topology (note: for clarity, not all interconnects are shown). b) Ready Test cell with three inverters and an AND gate c) N^{th} Order Priority Test cell.Each N^{th} order Priority Test cell contains an 8-bit comparator, an 8-bit 2:1 MUX, an (N-1)-bit 2:1 MUX, an OR gate, an AND gate, and a NAND gate (d) Event- Test cell

Figure2.ReferenceRTMEventand TimeManagementArchitecture

To eliminate small objects, connected component labelling is applied to the resultant image.(c) represents text detection by applying second set of criteria which eliminates all the objects whose area is less than 300 and filled area is less than 500.

IV. EXPERIMENTS

In order to formally justify the use of the Real-Time Task Manager inactualreal-time systems,an accurate quantification of the effects that it has onperformance is doneby analysing models of real-time systems that use RTOSes. All measurements are made on an in-house cycle-accurate C-based simulator of the Texas Instruments TMS320C6201. The
processoris a high-performance 32-bit VLIW fixed-point DSP with a 200 MHz clock that can issue eight 32-bit instructions per cycle [11]. The simulator is capable of loading the same executable binaryfiles that run in actual systems and executing them exactly asthey would on a real processor.
RTOS used in thisstudy: C/OS-II, a popular commercial pre-emptiveRTOS [6] In C/OS, semaphores areused to synchronize the access of the data channel by the two tasks. An additional noise task (with a period of 32 ms) purpose

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 6, Special Issue 14, August 2017

Two Days National Conference on Modernization in Engineering Science (NCMES 2K17)

23rd-24th March 2017

Organized by

Dept. of EEE, Sri Krishna College of Engineering & Technology, Coimbatore 641 008, Tamil Nadu, India

is to insert the type of background noise commonly present in real-time applications, e.g., many systems have LCD displays which have to be periodically refreshed. Aperiodic tasks which respond to external stimuli, such as I/O events are simulated as a by interrupts in MC/OS systems. The I/O events arrival times are geometrically distributed with an average arrival rate of 10 milliseconds.

C/OS-II: In Figure 4**,** The RTOS processor utilization has been divided into fivecategories for system configurations thatuse C/OS. These five categories include Scheduling, Interprocess Communication i.e. event Management, timer Interrupts, processing clock ticks i.e.timemanagementor updatingthe timer queue anda miscellaneous categorywhichincludes task Creation etc.

As seen in the graph infig. 4,the RTOS processorutilization without the RTMcan be quite significant. The bulk of the reduction comes from converting the overhead of time management operations thatreadytasks thatreach theirrelease times at every clock tick from a linear one to a constant one. The gains in event management and scheduling are lower, 14% and 30% respectively, because of the effectiveness of the software-based bit vector scheme employed in MC/OS. The timer interruptoverhead and miscellaneouscategory which are not optimized account for a constant less than 1% of the processor utilization, both with and without the RTM. The basicRTOS operations that theRTMimplements result inthe processing overhead required by C/OSto be reducedby 60% to 90%.

V. REAL-TIME TASK MANAGER IMPLEMENTATION

The RTM is a hardware module that implements a task database and thus supports the key RTOS operations: task scheduling,time management,and eventmanagement. It isan on-chip peripheral that communicates withthe core via a memory-mapped interface. The RTM is pictured in Figure 1; where each record contains information abouta singletask.The RTM is accessedthrough the global address space and each record can be read from or written toas ifitwere anyother array of structures.

A record is composed of four individually addressable fields. Each record contains a*status field*,containing several bitsthat describe the status of the corresponding record. The*valid bit* is necessary to indicate if that the record isusedby a task. The *delayed bit*indicates that the taskiswaiting forthe amountof clock ticks specified by the*delay field* before beingready-to- run. The event *bit*indicates that the task is pending on the event with the identifier specified by the event *ID field*. The *suspended bit* indicates that thetask hasbeen suspended. If the valid bit isset,but the delayed, event, andsuspended bits are clear, thenthe task is ready-to-run. Finally, the priority *field* indicates the task's priority. Also, the maximum number of records is some fixedconstant, such as 64 or 256. The RTOS can issue instructions toor retrievecertain information from the RTMvia a set of memorymappedregisters. The RTM can be queried for the highestpriorityreadytask and thus supports static-priority scheduling. The RTM does time management bydecrementing the delay fieldsof allrecordsby the numberof clock ticks specified by theRTOS and updating the delay bit if this value reaches zero. Event management is very similar tostatic-priority scheduling. TheRTM queries for the highest priority task that is pending on a given event identifier, instead of querying its data structure for the highest priority readytask. The software coding is in Turbo C for 1) scheduling 2) Event Management 3) Time Management and the timing analysis was calculated using Pentium Processor. The hardware design was designed and the VHDL coding was written for the hardware circuitry and downloaded using the XILINX-SPARTAN 2E FPGA kit. Results are as follows:

DESCRIPTION	HARWARE SIMULATION TIME	SOFTWARE SIMULATION TIME
Task scheduling	40.999 ns	5 micro s
Event Management	40.999ns	1.67 micro s
Time Management	5.977 ns	30 micro s

Table 1: Simulation result

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization *Volume 6, Special Issue 14, August 2017*

Two Days National Conference on Modernization in Engineering Science (NCMES 2K17)

23rd-24th March 2017

Organized by

Dept. of EEE, Sri Krishna College of Engineering & Technology, Coimbatore 641 008, Tamil Nadu, India

REFERENCES

[1]J.Adomat,J.Furunäs,L.Lindh,andJ.Stärner."Real- Time KernelinHardwareRTU:Asteptowardsdeterministicand highperformancereal-timesystems."In*ProceedingsofEighth EuromicroWorkshoponReal-Time Systems*, L'Aq- uila, Italy, June1996,pp.164-168.
[2] M. Bach.*TheDesignofthe UNIX Operating System*.Pren- tice-Hall, Englewood Cliffs,NJ, 1986.
[3] K. Baynes, C. Collins, E. Fiterman, B.Ganesh,P.Kohout, C. Smit, T. Zhang,andB. Jacob."ThePerformanceand ENergyConsumption ofThreeEmbeddedReal-Time Op- eratingSystems."In*Proceedingsofthe2001International Conference onCompilers,Architecture, andSynthesis for EmbeddedSystems (CASES '01)*, Atlanta, GA, November 2001,pp.203-210.
[4]R.Dick,G.Lakshminarayana,A.Raghunathan,andN.Jha."PowerAnalysisofEmbeddedOperatingSystems."In*Pro- ceedingsof the 37th DesignAutomationConference*, Los Angeles,CA,June2000.
[5]International TechnologyWorkingGroup.*International TechnologyRoadmapforSemiconductors2001Edition: ExecutiveSummary*.Semiconductor Industry Association,2001.
[6] J. Labrosse.*MicroC/OS-II: The Real-Time Kernel*.R&D Books, Lawrence, KS, 1999.
[7] C. Lee,M.Potkonjak, andW. Mangione-Smith. "Media- Bench:AToolforEvaluatingandSynthesizingMultimedia andCommunicationsSystems."In*Proceedingsofthe30th Annual InternationalSymposium onMicroarchitecture(MICRO '97)*,ResearchTriangle Park,NC,December 1997.
[8] J.Liu.*Real-TimeSystems*.PrenticeHall,UpperSaddleRiv- er, NJ, 2000.
[9] J. Mulder, N. Quach,and M. Flynn."AnArea Model for On-Chip Memories andits Application."*IEEEJournal of SOLID-STATE Circuits*,Vol.26,No.2,February1991,pp.98-106.
[10] TexasInstruments.*CodeComposerStudioUser's Guide*. February2000.
[11] J. Turley andH. Hakkarainen."TI's New 'C6xDSP Screamsat1,600MIPS."*TheMicroprocessorReport*,Vol.11, 1997,pp.14-17.
[12] V. J. Mooney and D.M.Blough,"A Hardware-Software Real-TimeOperatingSystemFrameworkforSOCs,"IEEE DesignandTestof Computers, pp. 44-51,November-De- cember2002.
[13] B. E. S. Akgul andV. J. Mooney,"The System-on-a-Chip Lock Cache," InternationalJournal of Design Automation forEmbeddedSystems,7(1-2)pp.139-174,September 2002.
[14] V. MooneyandG. DeMicheli,"Hardware/Software Code- signofRun-Time Schedulers forReal-TimeSystems,"*De- sign Automation of EmbeddedSystems*,6(1), pp. 89-144, September2000.
[15] T. Coopee."Embedded Intelligence." *InfoWorld*, Novem- ber 17, 2000.