N.Vignesh, Venkadesh.R, DR.K.Rajan / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 5, September- October 2012, pp.074-078

Simulation Of Pre-Emptive Scheduling For Accuracy Improvement By Using RTOS

N.Vignesh*, Venkadesh.R, DR.K.Rajan*****,**
*(Assistant Professor, Vel Tech Engineering college, Chennai-62.)
***(Vice-Principal, Vel Tech Engineering College Chennai-62, India.)
****(Principal, Vel Tech Engineering college, Chennai-62.)

ABSTRACT

Now-a-days Hardware and Software are developing very high level. But with the increasing of SOC designs, Hardware dependent Software (HDS) become Critical. In Previous work they introduced abstract RTOS modeling, which exposes dynamic scheduling effects early in the system design. However, such models insufficiently capture preemption. But the accuracy of preemption depends on the granularity of the timings. So we cannot take accurate readings. For an accurately modeled interrupt response time, very fine-grained timing annotation is necessary. It contradicts the RTOS abstraction idea and is detrimental to simulation performance.

In current modeling of abstract software execution (abstract RTOS on an abstract processor), preemption modeling highly depends on the timing annotation granularity. Scheduling decisions are made at the boundaries of wait-for-time statements. Hence, preemptive scheduling in an abstract model (e.g. after an interrupt) may be delayed by up to the longest time annotation in the whole application. Minimizing this error by using finer grained timing annotation, however, is undesirable due to a slower simulation with the dramatically increased number of wait-for-time statements, and the difficulty to obtain accurate fine-grained timing information. Therefore, preemption is inaccurately emulated in TLM, resulting in intolerable errors e.g. when simulating interrupt response times. In this paper, we introduce (simulation of) preemption in an abstract model and consequently improve dramatically the accuracy of the interrupt response time without increasing the number of wait-for-time statements.

Keywords –Pre-emptive Scheduling, RTOS, TLM-based Abstract RTOS, ROM, SLDL, Interrupt Latency.

I. Introduction

Current research work has addressed the increasing software content in modern MPSoC designs by utilizing software generation and abstract modelling of software. Abstract RTOS and processor Models have been proposed. They expose the effects of dynamic scheduling on a software processor already in early phases of the design. They have deemed crucial for design space exploration, e.g. for task distribution and priority distribution

However, current RTOS models poorly support pre-emption. An RTOS model executing in a discrete event simulation environment uses timing annotation to emulate target specific time progress (i.e. via wait-for-time statements). Scheduling decisions are made at the boundaries of these wait-for-time statements, very similar to cooperative multitasking. Hence, the accuracy of pre-emption depends on the granularity of the timing annotations (Figure 1).

A real CPU provides the finest granularity, checking at each clock cycle for incoming interrupts. Abstract models can annotate each C-instruction, basic block, function, or coarsely grained each task. However, accurate emulation of pre-emption requires fine grained annotation (e.g. at C statement level). On the other hand, using fine grained annotation has two drawbacks. It (a) slows down simulation speed, and (b) fine grained annotation information may not easily be available for a given application.

Figure 1. Granularity of timing annotation.

2. Problem Definition

In current modelling of abstract software execution (abstract RTOS on an abstract processor), pre-emption modelling highly depends on the timing annotation granularity. Scheduling decisions are made at the boundaries of wait for- time statements. Hence, pre-emptive scheduling in an abstract model (e.g. after an interrupt) may be delayed by up to the longest time annotation in the whole application. Minimizing this error by using finer grained timing annotation, however, is undesirable due to a slower simulation with the dramatically increased number of wait-for-time statements, and the difficulty to obtain accurate fine-grained timing information. Therefore, pre-emption is inaccurately emulated in TLM,

resulting in intolerable errors e.g. when simulating interrupt response time

3. Related Work

Abstract RTOS models have been developed that execute on top of System Level Design Languages (SLDLs) (e.g. SystemC, SpecC, proposes SoCOS, a high-level RTOS model. It interprets a proprietary language, describing RTOS characteristics, using a specialized simulation engine. Our proposed solution uses a standard unmodified discrete event simulator presents modelling of fixed priority pre-emptive multi-tasking systems. However, it uses SpecC specific concurrency and exception mechanisms and is limited in inter-task communication. In contrast, our proposed solution does not rely on SpecC specific primitives and provides task communication.

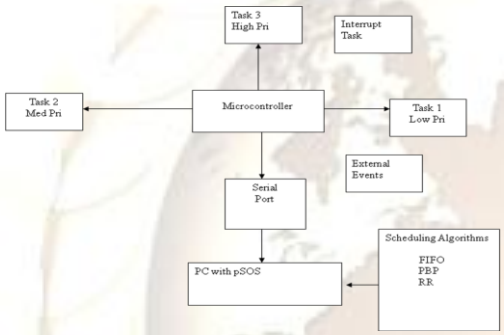

Figure-2. RTOS Block Diagram

It introduces abstract scheduling on top of SpecC, providing scheduling primitives found in a typical RTOS and allows modelling of target-specific execution timing. However, it emulates pre-emption only at the granularity of the timing annotation. In this paper, we will eliminate this restriction. It describes an RTOS centric co simulator, using a host compiled RTOS. However, it does not include target execution time simulation.

4. Abstract RTOS Modelling

We will _first describe a current approach of abstract RTOS modelling [6] (subsequently called TLM-based RTOS) and reveal the limitations in pre-emption modelling. Second, we will introduce the novel ROM-based abstract RTOS and show how it overcomes the TLM limitations.

4.1. TLM-based Abstract RTOS

The TLM-based abstract RTOS maintains a task state machine for each module/behaviour as shown simplified in Figure 2. Each action, which potentially changes scheduling, is wrapped to interact with the abstract RTOS model (e.g. task create, - suspend, - resume, semaphore acquire- release). For example, if a running task starts pending on a non-available semaphore, its state changes from RUN to WAIT (as in a regular RTOS). The abstract RTOS [6], keeps track of all task states and dispatches tasks using primitives of the underlying SLDL (e.g. events). It sequentializes the task execution according to the selected scheduling policy.

A pre-emption, as a result of an external interrupt, can occur at any point in time. Since a wait-for-time increases.

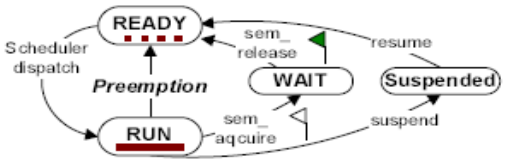

Figure-3. RTOS and Task Diagram

Simulation time and a pre-emption will occur while executing this statement. With the scheduling decision being made only at the end of the time increase, the pre-emption (dispatch of the selected ISR task) takes effect after the wait-for-time statement. This delays pre-emption scheduling and subsequently increases the latency for an ISR. Figure 3 shows a pre-emption situation handled by different approaches. We use line styles to indicate task states: a solid line represents RUNNING, dashed line READY and no line indicates the WAIT state. The empty flag indicates pending on a semaphore (or event) a filled flag its release.

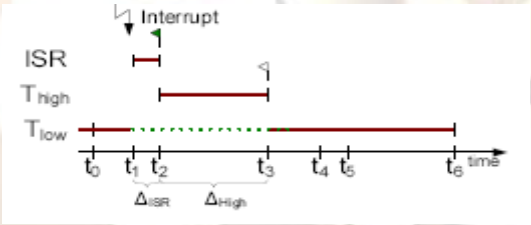

Figure-3(a) Processor

This Figure 3(a) depicts pre-emption on a real processor as a reference. While the low priority task T_{low} executes, an interrupt pre-empts at t1 and triggers the ISR. The ISR activates T_{high} at t2 and Finishes. T_{high} computes until t3 when acquiring a semaphore. Subsequently, the pre-empted T_{low} resumes and Finishes the section of computation at t6.

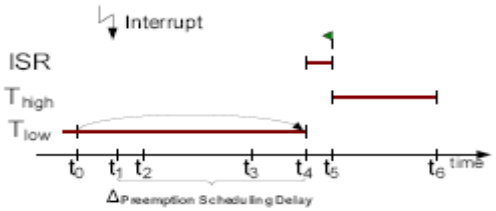

Figure- 3(b) TLM

This Figure 3(b) shows pre-emption in the TLM-based RTOS. The section executed by T_{low} is annotated with a single wait-for-time statement (from t0 to t4 depicted by an arc). Since the TLM-based RTOS evaluates scheduling at boundaries of wait-for-time statements, the interrupt occurring at t2, is evaluated only at t4. Then, it schedules first the ISR, then T_{high}. Note that the TLM-based RTOS is highly inaccurate. T_{high} finishes late at t6 (instead of t3). Analogous, T_{low} finishes early at t4 (instead of t6).

4.2. Result Oriented Modelling (ROM)

ROM is a general concept for abstract yet accurate modelling of a process that was demonstrated for communication modelling [16]. ROM assumes a limited observability of internal state changes of the modelled process. It is not necessary to show intermediate results of the process to the user, as in a .black box. approach. The only goal of ROM is to produce the *end result* of the process fast. Hiding of intermediate states gives ROM the opportunity for optimization. Often, intermediate states can be entirely eliminated. Instead, ROM utilizes an Optimistic predicts approach to determine the outcome (e.g. termination time and final state)

4.3. ROM-enhanced Abstract RTOS

Our ROM-enhanced abstract RTOS is based on the same principles as the earlier described TLM-based abstract RTOS. It extends all primitives, which potentially trigger scheduling, to interact with a centralized abstract RTOS model. However, ROM differs in the implementation of three crucial elements: (a) integration of interrupts, (b) wait-for-time statements, and (c) dispatch implementation. As a result, the ROM-based RTOS handles pre-emption With higher accuracy by allowing pre-emption of wait for time statements.

5. Analysis of Potential Benefits

In order to quantify the benefits of our ROM-based model, we first statically analyze five applications with function level timing annotations. In a second step, we measure one application.

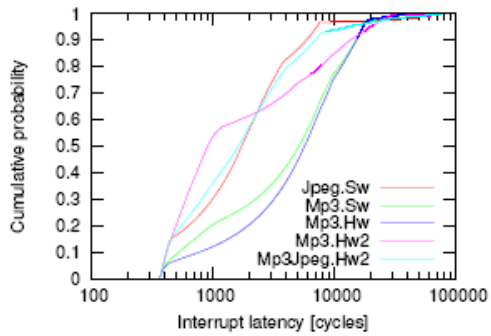

Figure-4. Statically analyzed TLM ISR latency.

More formally, the execution duration (busy time) of an application is captured in N wait-for-time statements, each annotates a duration of Wi and is executed Ci times. With this definition, the application execution time can be computed as Texec =PN; i=1; Ci =Wi. The probability P(Tdel) of incurring a delay of Tdel is then2:

Table 1. Statically analyzed TLM ISR latency.

	Specified	50%tile	96%tile	Max
Jpeg.Sw	366	1734	7434	75693
Mp3.Sw	366	5146	17750	55190
Mp3.Hw	366	5710	17810	55255
Mp3.Hw2	366	910	21538	38911
Mp3Jpeg.Hw2	366	1629	16291	75932

The pre-emption scheduling delay, thus the error in ISR latency, has a significant spread in a TLM. For our analyzed applications, 50% of ISRs will be delayed by up to 5,710 cycles. For the Mp3.Hw example 46% of ISRs will be delayed between 5,710 and 17,810 CPU cycles. In comparison to the specified delay of 366 cycles, our analysis indicates a potential improvement in the order of two magnitudes. In general, the actual improvement will depend on the application and its timing granularity.

6. Experimental Results

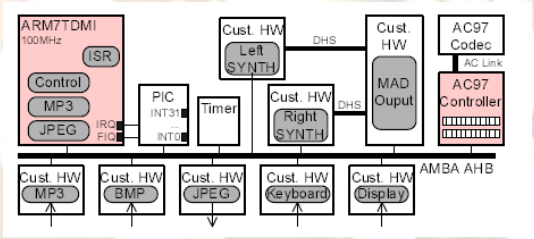

Figure-5 MP3 JPEG media

In order to demonstrate the benefits of our ROM-based abstract RTOS model on a real-world design example (Figure 7), we have implemented it on top of SCE [1] using the SpecC SLDL. We realized the ROM-based RTOS without any change in the simulation engine, it only uses standard primitives (events and wait-for-time statements). Note that the ROM concept is generic and can be directly applied to other SLDLs such as SystemC as well.

To measure the improvements, we use the ROM-based abstract RTOS in an industrial sized example as outlined in Figure 7. An ARM7TDMI running ¹C/OS-II [13] concurrently decodes a MP3 stream and encodes a JPEG picture. The processor is assisted by 3 HW accelerators, an additional set of HW units perform input and output. We focus in this simulation on the audio output. The ARM writes the decoded samples into the AC97 controller, which feeds them via an AC-Link to an AC97 codec [12].

Upon a half- filled FIFO, the AC97 controller triggers an interrupt to the ARM which then writes additional samples into the FIFO.

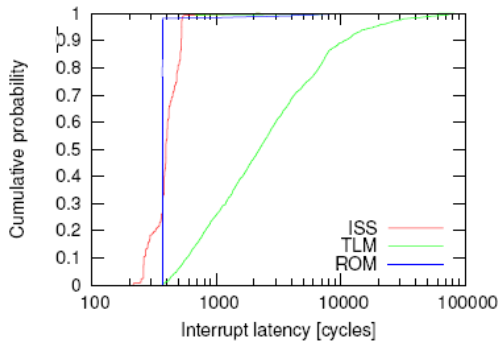

Figure-6 Measured Interrupt Latency

This Figure 6 shows the ISR latency for three solutions: execution on a cycle accurate ISS [2], simulation using the TLM-based RTOS, and using our ROM-based solution. The logarithmic x-axis denotes the ISR latency in CPU cycles. The y-axis denotes the cumulative probability. As an example, the TLM line reads 0.41 at 1000 cycles, indicating that 41% of the ISR invocations will be delayed by 1000 cycles or less. Table 2 shows the same data in numerical form.

Table 2. Measured interrupt latency.

	Min	Avg	50% tile	96% tile	Max
CPU	213	403	392	525	2247
TLM	367	5062	2218	21298	81789
ROM	366	436	367	368	9744

Our ROM-based abstract RTOS model, on the other hand, shows a very tight distribution. The minimal latency and the 96th percentile are only 2 cycles apart. Interestingly, the maximum observed latency reached almost 10,000 cycles. Here, the interrupt occurred while the CPU just started another ISR. Since both interrupts use the same priority level, no pre-emption occurred and the measured ISR started late. The ROM ISR latency distribution matches the CPU within 8% in terms of average and 50th percentile. At this point, we do not model RTOS critical sections, hence ROM does not show the same variation the CPU does.

7. Conclusion

In this Paper, we have presented a novel approach for modeling preemption in an abstract RTOS model. Our solution is based on the Result Oriented Modeling technique, previously applied only to communication modeling. While a TLM-based RTOS model relies on fine-grained timing annotations to emulate preemptions, our ROM-based model allows accurate preemption at any point. ROM significantly increases the timing accuracy of preemption simulation without demanding fine-grained timing information and without reducing simulation performance. With this accuracy improvement, ROM is an enabler to further expand the use of abstract modelling. This work is the first to show that the ROM concept is applicable outside of the communication domain. Where in communication modelling it is tied to a particular bus model, here the ROM approach is not application specific. Any application scheduled on the ROM-based RTOS will benefit from the enhanced accuracy.

References

[1] T. Gr¨otker, S. Liar, G. Martin, and S. Swan. *System Designwith SystemC*. Kluwer Academic Publishers, 2002.

[2] Z. He, A. Mok, and C. Peng. Timed RTOS Modelling for Embedded System Design. In *RTAS*, San Francisco, 2005.

[3] K. Hines and G. Borriello. A Geographically Distributed Framework for Embedded System Design and Validation. In *DAC*, San Francisco, CA, June 1998.

[4] S. Honda et al. RTOS-Centric Hardware/Software Cosimulator for Embedded System Design. In *CODES+ISSS*, Stockholm,Sweden, Sept. 2004.

[5] Y. Hwang, S. Abdi, and D. Gajski. Cycle Approximate Retargettable Performance Estimation at the Transaction Level. In *DATE*, Munich, Germany, Mar. 2008.

[6] Intel Corporation. *Audio Codec '97 Component Specification*, Sept. 2000.

[7] H. Posada et al. RTOS modelling in SystemC for real-time Embedded SW simulation: A POSIX model. *Design Automation For Embedded Systems*, 10(4):209.227, Dec. 2005.

[8] G. Schirner and R. D¨omer. Result Oriented Modeling a Novel Technique for Fast and Accurate TLM. *IEEE TCAD*,1688.1699, Sept. 2007.

[9] H. Tomiyama et al. Modeling fixed-priority pre-emptive multi-task systems in SpecC. In *SASIMI*, Nara, Oct. 2001.

N.Vignesh @ Selvaganapathy received B.Tech Degree in 2007 in Electrical and Electronics Engineering from Sri Manakula Vinayagar Engineering College affliated to Pondicherry University and M.E Embedded System Technolgies degree received in the year 2009 from Vel Tech Engineering College affliated to Anna University chennai. He has 3 years of teaching experience in engineering colleges. His Research are interest in Speed Control of Electrical Machine, Switching Converter Technolgy and Drives. He has published 2 international Journals, 2 International conferences and attended Faculty Development programmes. He

is currently working as a Assistant Professor in the Department of Electrical and Electronics Engineering, Vel Tech Engineering college, Chennai-62, India. .

Venkadesh.R completed his B.E. Computer Science and Engineering from University of Madras in 1999 and M.Tech Electronics and Communication Engineering from Pondicherry University in 2004. He has 13 years of teaching experience in engineering colleges. He has published 2 international Journals, 3 International conferences, attended workshops and Faculty Development programmes. Interested area is Communication and Networking. At present he is working as Vice-Principal and Assistant Professor, Department of Electronics and Communication Engineering, Vel Tech Engineering college, Chennai-62, India.

K.Rajan completed his B.E. Electrical and Electronics and M.E. Applied Electronics at **P.S.G. College of Technology,** Coimbatore. He has completed his Ph.D. from J.N.T.U. Anantapur, AP in high frequency power system for industrial and commercial zones. He has 19 years of industrial experience in R&D, production and 9 years of teaching experience in engineering colleges. He has published 4 international and national journals. He also presented papers in 14 international and national conferences. Interested area is power electronics and power system. At present he is working as Principal in Vel Tech Engineering College. – RS Trust.