一种基于宿主机/目标机架构的追踪/重演方法*

陈 艳[1]，徐晓峰[2]，汤伟宾[1]，郭东辉[1]
(1 厦门大学信息科学与技术学院 厦门 361005; 2 厦门大学物理系 厦门 361005)

摘 要：实时嵌入式系统固有的不确定性使得系统运行具有不可重现性，从而造成系统调试与测试时故障可能无法重现。提出一种基于宿主机/目标机架构的追踪/重演方法来解决实时嵌入式系统运行的不可重现性问题。该方法通过插装探针来追踪系统的任务调度、任务间通信同步以及 I/O 操作等信息，并自动将系统的执行信息保存到宿主机端，然后通过任务控制模块来控制系统中的任务按照原有的先后顺序来执行，从而实现实时嵌入式系统执行情况的正确回放。目前，该方法已在 ML505 开发板和 uC/OS-II 操作系统上进行实现，并已成功应用到 IC 图像拍摄系统中。通过实验分析表明，该方法能够以较小的时间和空间开销实现实时嵌入式系统运行情况的追踪和重演。

关键词：实时嵌入式系统；不确定性；追踪/重演；宿主机/目标机
中图分类号：TP316 **文献标识码**：A **国家标准学科分类代码**：520.40

Record/replay approach based on host/target architecture

Chen Yan[1], Xu Xiaofeng[2], Tang Weibin[1], Guo Donghui[1]
(1 School of Information Science and Technology, Xiamen University, Xiamen 361005, China;
2 Department of Physics, Xiamen University, Xiamen 361005, China)

Abstract: Due to the non-determinacy of real-time embedded system, some bugs in the system cannot be reproduced during debugging and testing. A record/replay approach based on host/target architecture is presented to deal with the non-reproducibility problem of the executions of real-time embedded system. In the approach, context switch events, inter-task communication and synchronization events and I/O operation events are all instrumented. During the execution of the system, the related events and operations are traced and the execution information is saved into the host machine. Then, a task control module is designed to guarantee the execution of tasks in the original order so that the recorded execution information can be replayed correctly. The approach has been implemented in the ML505 board and uC/OS-II, and it has been applied to an IC image shooting system successfully. Finally, case study shows that the approach can record and replay the execution of real-time embedded system with small time and space overhead.

Key words: real-time embedded system; non-determinacy; record/replay; host/target

1 引 言

实时嵌入式系统在任务调度、任务间通信同步以及外界环境等方面的不确定性使得系统运行具有不可重现性(non-reproducibility)[1]。即使在相同的输入条件下，由于突发中断、随机延时以及噪声等影响，系统相继执行不能保证再现先前的执行过程，从而导致系统调试或测试时故障可能无法重现[1]。因此，为了解决实时嵌入式系统固有的不确定性因素可能带来的系统可靠性问题，在开发实时嵌入式系统过程中，需要考虑如何实现系统调试与测试过程中的故障重现。

在并行软件系统开发过程中，通常采用追踪/重演方法[2-14]来实现软件调试与测试时的故障重现，即通过对目标系统的运行情况进行追踪，并记录相关的执行信息，之后在相同输入条件下，根据追踪记录下来的执行信息来执行系统先前的状态，以达到系统运行情况的再现重演。如文献[2−3]是针对 Solaris 操作系统执行并行程序采

收稿日期：2009-04 Received Date：2009-04
*基金项目：国家自然科学基金(60753001)、教育部新世纪人才计划项目资助

用追踪/重演方法来实现测试故障重现技术的；文献[4−5]则是采用相同的追踪/重演方法来实现 Java 多线程程序的测试故障重现；文献[6-8]也介绍了如何采用追踪/重演方法来调试基于消息传递的并发程序；文献[9]介绍了一种基于事件追踪方式的 MPI 并行程序性能软检测及可视化方法。但是这些追踪/重演方法都是针对在普通计算机系统上运行的并行程序，不适用于实时嵌入式系统调试与测试的故障重现。这是由于通常情况下实时嵌入式系统的硬件资源有限，无法直接存储大量的追踪信息；而且实时嵌入式系统的实时中断及任务实时性要求也不同于普通计算机系统，在追踪/重演系统执行信息时还需要考虑任务调度及任务间通信同步等不同情况。另外，尽管也有文献[10-14]介绍通过增加硬件辅助设备来实现嵌入式系统的故障重现技术，但其实现方法比较困难。

为此，本文提出一种基于宿主机/目标机架构的追踪/重演方法来解决实时嵌入式系统运行的不可重现性问题以实现系统调试与测试时的故障重现。该方法通过插装探针来追踪系统的任务调度、任务间通信同步以及 I/O 操作等信息，并自动将系统的执行信息保存到宿主机端，然后通过任务控制模块来控制系统按照原有的执行状态运行，从而实现实时嵌入式系统执行情况的重新回放。

2 实时嵌入式系统的不确定性

实时嵌入式系统的任务包括系统级的任务(简称系统任务)和用户级的任务(简称用户任务)。其中系统任务是指实时操作系统内部本身自带的任务，如：在 uC/OS II 操作系统中自带 2 个系统任务 OSTaskStat 和 OSTaskIdle[15]，其中 OSTaskStat 用来统计运行时间，而 OSTaskIdle 则是在没有其他任务进入就绪时投入运行；用户任务是指用户根据实际需求编写的应用程序中所包含的任务。在实时嵌入式系统中，应用程序通常包含有多个用户任务，且用户任务之间通常采用基于优先级可抢占的调度算法[16]进行任务调度，并采用信号量(semaphore)和消息队列(message queue)等[15-16]机制进行通信和同步。由于实时嵌入式系统的运行环境复杂，外部事件或中断的发生时间与外界环境或用户操作相关，具有随机性，且在某些应用中，用户任务本身也含有一些随机延时操作，这些因素使得任务的执行时间和响应时间发生变化，从而导致系统任务调度及任务间通信同步具有不确定性，甚至导致竞态条件[2,6]的发生。

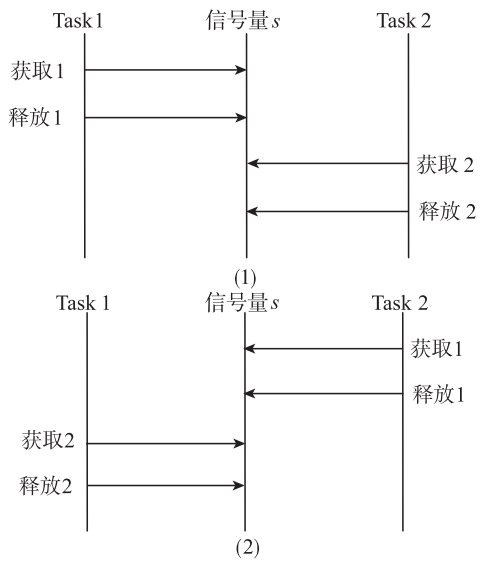
图 1 所示是一个实时嵌入式系统不确定性的实例。在该例子中，两个任务同时对共享资源进行操作并采用信号量进行同步，如图 1(a)所示（其中 semTake 表示获取信号量，semGive 表示释放信号量）。然而，在系统重复运行的过程中，可能存在两种不同的操作顺序，即可能是任务 1 先获取到信号量，任务 2 后获取到信号量，也可能是任务 2 先获取到信号量，任务 1 后获取到信号量，如图 1(b)所示。这两种不同的操作顺序可能造成系统不同的运行结果。因此，在系统开发、调试和测试过程中，必须采用有效的追踪/重演方法来解决系统任务调度及任务间通信同步的不确定性，以实现系统的故障重现。

```
Binary Semaphore s = available;

Task 1                              Task 2
……                                  ……
semTake(s);                         semTake(s);
//Access the critical resource      //Access the critical resource
semGive(s);                         semGive(s);
……                                  ……
```

(a)一个实时嵌入式系统的应用程序

(a)An application of real-time embedded system



(b)两种不同的操作顺序

(b)Two different operating orders

图 1 实时嵌入式系统不确定性的实例

Fig.1 An example of non-determinacy of real-time embedded system

3 基于宿主机/目标机架构的追踪/重演方法

本方法的实现方案主要体现在宿主机端和目标机端的设计，如图 2 所示。其中宿主机端由实时嵌入式系统开发工具及追踪/重演控制程序组成，本文主要是设计追踪/重演控制程序；目标机端则是由目标代理与驱动程序、实时嵌入式操作系统、追踪/重演组件程序及应用程序组成，而本文需要设计的是追踪/重演组件程序。追踪

/重演组件程序与追踪/重演控制程序之间采用以太网接口或串口通信，而实时嵌入式系统开发工具与目标代理程序之间则是通过嵌入式系统自带的串口或 JTAG 接口进行通信。

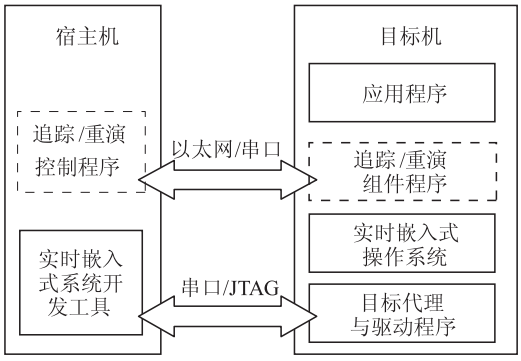


图 2　追踪/重演方法的总体设计图

Fig.2 Architecture of the record/replay approach.

图 3 和图 4 分别给出了追踪和重演的实现流程图，其中用虚框表示的模块是追踪/重演方法实现的关键模块。各个模块的功能如下：

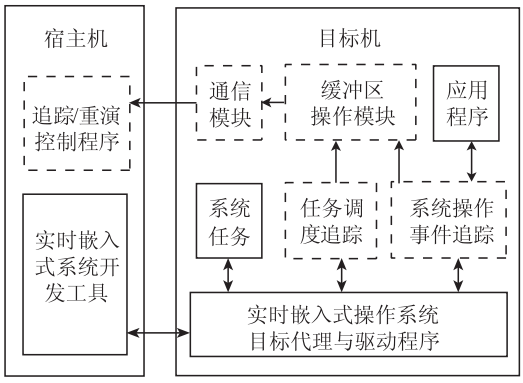


图 3　追踪的实现流程图

Fig.3 Control flow of recording

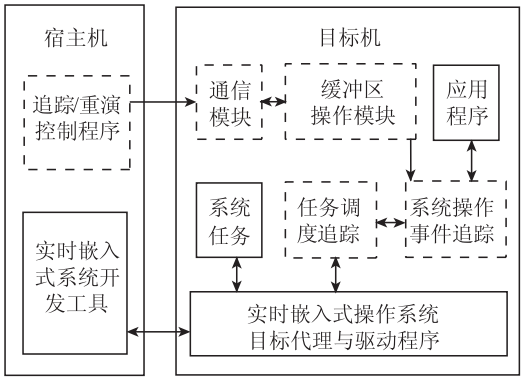


图 4　重演的实现流程图

Fig.4 Control flow of replaying

1)通信模块：在追踪和重演的过程中，该模块单独一个任务在运行，主要负责与位于宿主机端的追踪/重演控制程序进行通信和数据传输。在追踪时，若系统缓冲区不足，则需要调高该任务的优先级以便能将缓冲区中的数据及时传回宿主机端。

2)缓冲区操作模块：该模块的主要包括缓冲区的读写操作。

3)任务调度追踪：该模块负责追踪系统的任务调度信息。

4)系统操作事件追踪：该模块负责追踪应用程序中的系统操作事件，主要包括：获取信号量、释放信号量、发送消息和接收消息等任务间通信同步事件以及 I/O 读写操作事件。

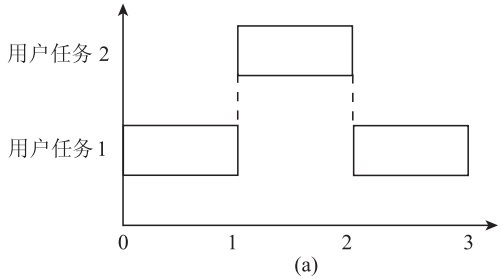
5)任务控制模块：在重演时，该模块负责处理先前追踪保存下来的任务调度信息，使用户任务按照原有的顺序执行。

6)系统操作事件重演：该模块负责重新回放先前追踪保存下来的执行信息。

7)追踪/重演控制程序：该程序位于宿主机端，其功能主要有：启动和停止系统的追踪或重演；在追踪时负责接收执行信息并保存到追踪文件；在重演时传输先前保存的执行信息到目标机端进行重新回放。

3.1　任务调度的追踪/重演

在实现系统任务调度信息的追踪时，追踪/重演组件程序通过任务切换钩子函数（task switch hook function）[15-16]将任务切换的探针插装到系统的内核中，从而保证系统每次进行任务调度时都能自动探测有关的任务切换信息，并传给宿主机端的追踪/重演控制程序进行保存。当系统进行任务切换时，若新任务是一个用户任务且与前一个用户任务不同，那么追踪/重演组件程序将记录此次任务切换的时间、新任务的名称、ID 以及优先等级等相关信息。图 5 展示了 3 种可能的任务切换场景。其中在图 5(a)中，时刻 1 和时刻 2 的任务切换都发生在用户任务之间，因此追踪/重演组件程序分别在这两个时刻记录任务切换的相关信息；在图 5(b)中，系统在时刻 1 从用户任务 1 切换到系统任务去执行，在时刻 2 又重新切换到用户任务 1 去执行，因此追踪/重演组件程序不保存任何的任务切换信息；在图 5(c)中，系统在时刻 1 从用户任务 1 切换到系统任务去执行，在时刻 2 切换到用户任务 2 去执行，在这种情况下，追踪/重演组件程序保存时刻 2 的任务切换信息。



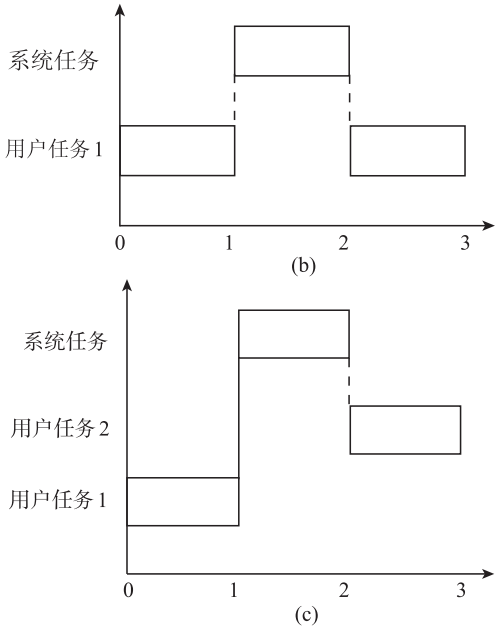
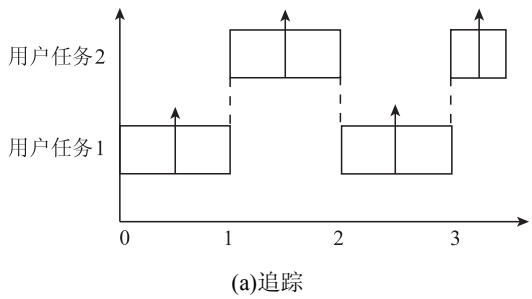


图 5 任务切换的 3 种可能的场景

Fig.5 Three possible scenarios of task switch

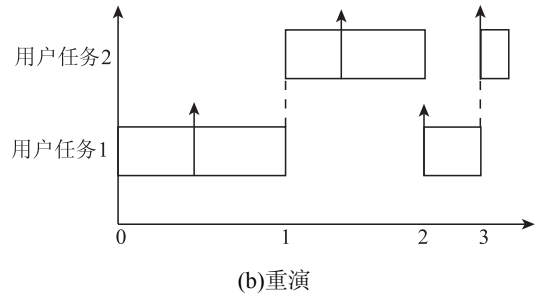
在调用所保存的执行信息进行重演时，追踪/重演组件程序中的任务控制模块按照调用的任务切换信息来挂起(suspend)正在运行的任务，并将系统切换到下一个用户任务去执行。

如图 6 所示是一个系统任务调度的追踪与重演实现例子(箭头“↑”表示系统操作事件)。在追踪系统执行信息时，追踪/重演组件程序保存用户任务 1 和用户任务 2 的切换信息以及系统操作事件的执行信息；当重演系统执行信息时，任务控制模块根据时刻 0 的任务切换信息让用户任务 1 首先执行，当任务 1 执行完第 1 个操作事件后需要执行第 2 个操作事件时，由于接下去的执行信息是任务切换信息，所以任务控制模块挂起用户任务 1，并将系统切换到用户任务 2 去执行；同样，当用户任务 2 执行完第 1 个操作事件后需要执行第 2 个操作事件时，也由于接下去的执行信息是任务切换信息，所以任务控制模块挂起用户任务 2，并又将系统切换到用户任务 1 去执行。这样就可以保证系统任务执行的先后顺序，从而实现系统执行信息的重演。



(a)追踪

(a) Recording



(b)重演

(b) Replaying

图 6 任务调度追踪/重演实例

Fig. 6 A record/replay example of task scheduling

3.2 系统操作事件的追踪/重演

系统操作事件主要包括任务间通信同步事件，即获取信号量、释放信号量、接收消息、发送消息等，以及 I/O 读写操作事件。这些事件信息追踪/重演的实现过程可以按图 7 所示来设计。在对这些事件信息进行追踪时，追踪/重演组件程序是通过对这些事件执行函数进行探针插装，当应用程序执行相应事件时，将自动探测相关的信息，如操作时间、函数参数、返回代码，出错信息等，并传给宿主机端的追踪重演控制程序进行保存。当对所保存的执行信息进行重演时，若从缓冲区中读取到的执行信息与该事件相匹配，则将该信息直接反馈给系统和应用程序；若读取到的执行信息是任务切换信息，则将其发送给任务控制模块，同时所在任务被挂起；若读取到的执行信息是其他类型的信息，如异步信号等，则对其进行相应的特殊处理。

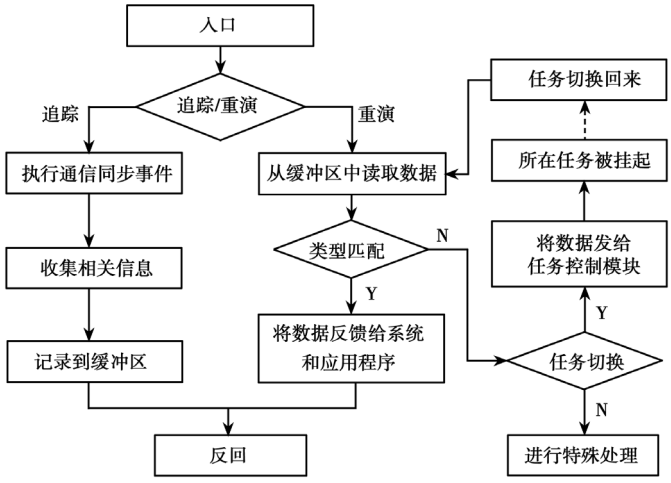


图 7 任务间通信与同步事件的追踪/重演

Fig.7 Record/replay of inter-task communication and synchronization events

4 实验与性能分析

目前，本文所阐述的追踪/重演方法已经在 Xilinx 公

司的 ML505 开发板上进行兑现。该开发板含有一个 MicroBlaze 嵌入式软核、一个 10/100 M 以太网接口以及 RS232 串口等资源。目标机采用 uC/OS II 实时操作系统[15]；宿主机采用 Windows XP 操作系统。目标机和宿主机之间使用串口和 JTAG 接口进行通信。

由于本文的追踪/重演方法是通过插装探针的方式来实现，因此该方法也会有时间和空间损耗。通过对 uC/OS II 实时操作系统中的任务切换(OS_TASK_SW)、获取信号量事件(OSSemPend)、释放信号量事件(OSSemPost)、发送消息事件(OSQPost)和接收消息事件(OSQPend)等操作进行性能测试，获得追踪/重演方法的平均时间和空间损耗如表 1 所示(目标系统的时钟频率为 100 MHz)。

表 1　追踪/重演方法中系统操作事件的时间和空间损耗情况

Table 1 Time and space overhead of operating events in the record/replay approach

系统事件	原来/μs	插装后/μs	时间损耗/μs	内存损耗/B
OS_TASK_SW	3.000	5.500	2.500	24
OSSemPend	1.167	4.333	3.166	20
OSSemPost	0.833	4.167	3.334	16
OSQPost	0.833	4.333	3.500	16
OSQPend	1.333	5.333	4.000	36

在表 1 中，OS_TASK_SW 事件是统计系统 10 000 次任务切换所需时间的平均值，其计算方法可参考文献[16]。OSSemPend 事件和 OSSemPost 事件是分别统计 60 000 次信号量操作的平均时间损耗。OSQPost 事件和 OSQPend 事件也是分别统计 60 000 次消息队列操作的平均时间损耗，且这两个事件的时间和空间损耗与消息的长短有关：消息越长，时间和空间损耗也越大(在本次测试中，消息的长度为 18 个字节)。

此外，本文还将追踪/重演方法应用到一个 IC 图像拍摄系统中[17]中以验证其可行性。该系统主要由摄像头、显微镜、载物台、步进电机、步进电机驱动模块、ML505 开发板以及 PC 等部件组成，如图 8 所示。其中，PC 上安装有图像拍摄软件，可通过摄像头和显微镜来拍摄载物台上的 IC 芯片；系统将载物台安装到一个三维平动装置上，并通过 3 个步进电机来控制其运动；系统通过 ML505 开发板上的按键来控制步进电机的运动方向和步数。

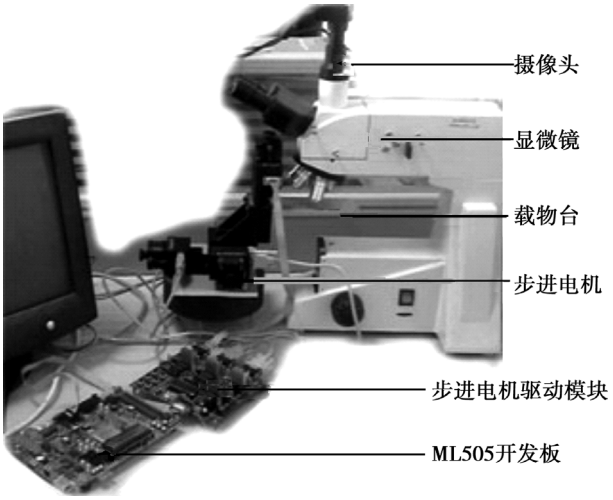
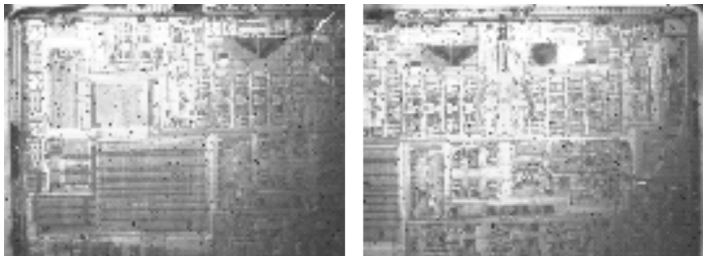


图 8　IC 图像拍摄系统

Fig.8 IC image shooting system

该系统的控制软件主要是运行在 ML505 开发板上，它是采用 uC/OS-II 实时操作系统运行 2 个任务。其中任务 1 负责实时采集开发板上的按键信息，并将信息传给任务 2；任务 2 负责对按键信息进行分析处理，并通过驱动模块控制步进电机进行相应的运动。由于开发板的按键输入是人为随机的，所以整个控制器的任务调度及任务间通信同步具有不确定性，从而使得步进电机的运动情况也具有不确定性，系统在运行调试过程中难以重复控制。

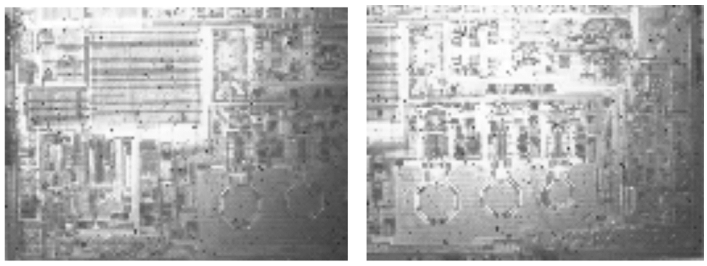
只要在 ML505 开发板上加入追踪/重演组件程序就可以实现对系统任务调度、按键操作(即 I/O 操作)以及任务间通信同步等执行信息的追踪并可将信息传输到 PC 端保存，而且在重演时可将执行信息传回到开发板中进行重新回放。因此，如果追踪/重演方法有效，那么系统在追踪和重演状态下步进电机的运动轨迹将是一致的，即在重演状态下系统所拍摄出的图像与追踪状态下所拍摄的图像一致。图 9 和图 10 分别是系统在追踪和重演状态下拍摄到的某款 IC 芯片的全局图像，其中(a)~(d)分别是该芯片左上、右上、左下、右下部位的图像。通过对比可知这两组图像高度一致，说明本追踪/重演方法可以解决该系统运行的不可重现性问题。



(a)　　(b)

图 9　系统在追踪状态下拍摄的图像

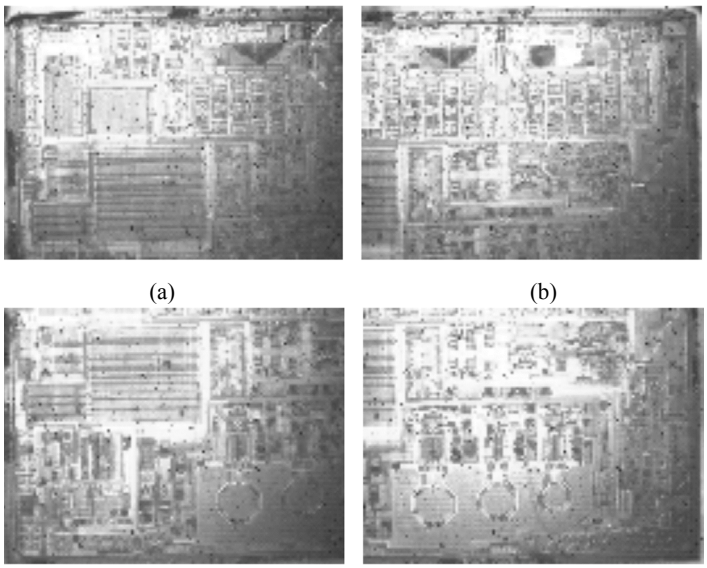
Fig.9 Images got during recording



(c) (d)

图 9 系统在追踪状态下拍摄的图像

Fig.9 Images got during recording



(a) (b)

(c) (d)

图 10 系统在重演状态下拍摄的图像

Fig.10 Images got during replaying

另外，表 2 是这 4 次拍摄过程中收集到的实验数据。其中每组数据都包含了 3 个步进电机总的运动步数、追踪文件的大小以及重演时间。如果将这 4 组的平均重演时间 33.516 s 除以平均运动步数 570，可得步进电机每动一步大概需要 59 ms 的重演时间(该时间包括重演时为了确保步进电机不丢步而加的延时)。由于系统采用手动方式进行 IC 图像拍摄，所以整个拍摄过程的时间较长，与之相比，本方法的重演时间开销较小，并可以准确地回放先前的拍摄过程。

表 2 实验数据统计结果

Table 2 Experimental result

拍摄次序	运动步数	追踪文件大小/B	重演时间/s
1	450	2 064	26.156
2	610	2 904	35.672
3	640	3 364	38.078
4	580	2 884	34.156
平均	570	2 804	≈33.516

5 结 论

本文提出了一种基于宿主机和目标机架构的追踪/重演方法，能够以较小的时间和空间开销实现实时嵌入式系统运行情况的追踪记录和重新回放，给实时嵌入式系统的开发与调试带来很大的方便。目前，该方法已在 ML505 开发板和 uC/OS-II 实时操作系统上实现，并成功地应用到 IC 图像拍摄系统中，解决了系统故障无法重现的问题。在接下去的工作中，会将本方法应用到其他硬件平台以及其他实时操作系统上去。

参考文献

[1] MCDOWELL C E, HEMBOLD D P. Debugging concurrent programs[J]. ACM Computing Surveys, 1989,21(4): 593-622.

[2] RONSSE M, DE BOSSCHERE K, CHRISTIAENS M, et al. Record/replay for nondeterministic program executions[J]. Communications of The ACM, 2003,46(9):62-67.

[3] RONSSE M, DE BOSSCHERE K. RecPlay: a fully integrated practical record/replay system[J]. ACM Transactions on Computer Systems, 1999,17(2):133-152.

[4] CHOI J D, SRINIVASAN H. Deterministic replay of Java multithreaded applications[C]. Proceedings of the SIGMETRICS symposium on Parallel and Distributed Tools, Welches, Oregon, United States, Aug. 1998:48-59.

[5] BOWEN A. A perturbation-free replay platform for cross-optimized multithreaded applications: IBM research report [R]. Sep. 2000.

[6] NETZER R H B, MILLER B P. Optimal tracing and replay for debugging message-passing parallel programs[C]. Proceedings of the ACM/ONR workshop on Parallel and distributed debugging, San Diego, California, United States, May. 1993:1-11.

[7] SAITO Y. Jockey: a user-space library for record-replay debugging[C]. Proceedings of the sixth international symposium on Automated analysis-driven debugging, Monterey, California, United States, Sep. 2005:69-76.

[8] 殷贤亮,丁宁. 基于插桩技术的并行程序的重演方法[J]. 华中科技大学学报, 2006,34(9):7-10.

YIN X L, DING N. Instrumentation-based replay of message-passing parallel programs[J]. Journal of Huazhong University of Science and Technology, 2006, 34(9):7-10.

[9] 刘晓平,卫兴武. 并行程序性能检测及可视化[J].仪器仪表学报, 2008,29(9):1831-1835.
LIU X P, WEI X W. Performance measurement and visualization for parallel program[J]. Chinese Journal of Scientific Instrument, 2008,29(9):1831-1835.

[10] MONTESINOS P, CEZE L, TORRELLAS J. DeLorean: recording and deterministically replaying shared-Memory multiprocessor execution efficiently[C]. Proceedings of the 35th International Symposium on Computer Architecture, Beijing, China, Jun. 2008:289-300.

[11] TSAI J, FANG K Y, CHEN H Y, et al. A noninterference monitoring and replay mechanism for real-time software testing and debugging[J]. IEEE Transactions on Software Engineering, 1990,16(8):897-916.

[12] XU M, BODIK R, HILL M D. A “flight data recorder” for enabling full-system multiprocessor deterministic replay[C]. Proceedings of the 30th annual international symposium on Computer architecture, San Diego, California, United States, Jun. 2003:122-135.

[13] XU M, HILL M D, BODIK R. A regulated transitive reduction (RTR) for longer Memory race recording[C]. Proceeding of the ASPLOS Conference, San Jose, California, United States, Oct. 2006:49-60.

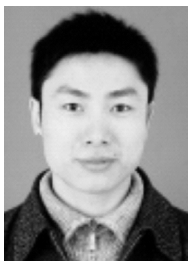
[14] NARAYANASAMY S, PEREIRA C, CALDER B. Recording shared memory dependencies using strata[C]. Proceeding of the ASPLOS Conference, San Jose, California, United States, Oct. 2006:229-240.

[15] LABROSSE J J. Micro C/OS-II The Real-time kernel [M]. 2nd ed. CMP Books, 2002.

[16] 罗蕾. 嵌入式实时操作系统及应用开发[M]. 北京：北京航空航天大学出版社, 2005.
LUO L. Embedded real-time operating system and applications development [M]. Beijing: BUAA Press, 2005.

[17] TANG W B, LIAO Y H, CHEN Z C, et al. Auto-focusing system for microscope based on computational verb controllers[C]. Proceeding of 2nd International Conference on Anti-counterfeiting, Security and Identification, Guiyang, China, Aug. 2008:84-87.

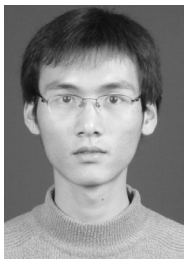
作者简介



陈艳，2004 年于厦门大学获得学士学位，现为厦门大学信息科学与技术学院博士研究生，并于 2007—2008 年期间赴美国 UTD 大学和 ASU 大学做交流访问学者，主要研究方向为实时嵌入式系统的分析与测试。

E-mail: yanchen@xmu.edu.cn

Chen Yan received his BSc from Xiamen University in 2004. Now he is a PhD student in School of Information Science and Technology, Xiamen University. He was a visiting scholar in UTD and ASU from 2007—2008. His research interests are real-time embedded system analysis and testing.



徐晓峰，2005 年于厦门大学获得学士学位，现为厦门大学物理系博士研究生，主要研究方向为软件测试与调试。

E-mail: xxfeng@xmu.edu.cn

Xu Xiaofeng received his BSc from Xiamen University in 2005. Now he is a PhD student in Department of Physics, Xiamen University. His research interest is software testing and debuging.

郭东辉（通讯作者），1994 年获厦门大学半导体物理与器件物理专业博士学位，现任厦门大学信息科学与技术学院教授、博士生导师兼副院长，主要研究方向为人工智能、网络通讯、软硬件协同设计等。

E-mail: dhguo@xmu.edu.cn

Guo Donghui(Corresponding author) received his PhD degree in Semiconductor Device & Device Physics from Physics Department of Xiamen University in 1994. He is now a professor, doctoral supervisor and vice-principal of the School of Information Science & Technology of Xiamen University. His research interests in artificial intelligence, network communication, and software-hardware co-design and so on.