第 33 卷 第 5 期 计 算 机 工 程 2007 年 3 月
Vol.3 No.5 Computer Engineering March 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)05—0246—03 文献标识码：A 中图分类号：TN911.72

一种快速断点仿真器的软硬件协同设计

陈东晓，高 磊，梅优良
(浙江大学信息与电子工程学系，杭州 310027)

摘 要：仿真器是利用硬件在仿真环境中对程序进行仿真、调试的重要工具。在断点执行方式下，需要检查控制断点和数据断点，仿真器难以保持较快的仿真速度，提出了一种软硬件协同的快速断点仿真器设计方案，在仿真过程中有效地简化了目标程序和监控程序的交互过程，从而可以获得较快的仿真速度，并简化了电路的设计。

关键词：仿真器；控制断点；数据断点；软硬件协同设计

Software/Hardware Co-design of Fast Break-point Emulator

CHEN Dongxiao, GAO Lei, MEI Youliang
(Dept. of Information Science and E1ectronic Engineering, Zhejiang University, Hangzhou 310027)

【Abstract】 Emulator is an important tool for emulate and debug programs in an emulational environment provided by hardware. Because of checking the break points and the watch points, it's hard to keep the emulator in high speed under break point mode. This paper introduces a software/hardware co-design method to simplify the intercommunication of target program and monitor program so that the emulating speed is increased and the circuit is simplified.

【Key words】 Emulator; Break point; Watch point; Software/Hardware co-design

仿真器(emulator)是一种利用硬件对目标程序进行仿真、调试的工具。相对于模拟器(软件仿真器，simulator)，仿真器具有仿真速度快和环境真实等优点。仿真器还可以用来对正在开发的目标芯片进行验证，在较真实环境中及时发现问题。但是仿真器也有开发周期长、硬件成本高等缺点。仿真器允许用户在程序中设置控制断点和数据断点，使得用户可以很方便地对程序进行调试。但由于需要检查控制断点(break point)和数据断点(watch point)，仿真器往往难以达到较快的仿真速度。

本文提出了一种软硬件协同设计的方法，将控制断点的检查限制在目标程序中，并利用链接器控制了数据断点的检查条件，在仿真过程中有效地避免了目标程序和监控程序的过多交互，可以较大幅度提高断点仿真器的执行速度，并简化电路的设计。

1 快速断点仿真器的设计原理

仿真系统由上位机和下位机组成。上位机(主机)一般是普通计算机或嵌入式系统，用来提供用户交互环境和对下位机的控制。上位机的软件系统一般包括集成界面、编译器、汇编器、链接器和装载器等。下位机(又称作评估板)是插有仿真目标芯片的开发板，一般由控制芯片、控制电路、目标芯片、通信接口和存储器等组成。上位机通过通信接口和下位机相连；待仿真的目标程序在目标芯片上执行；控制芯片和控制电路利用监控程序与上位机进行交互并监控目标芯片的执行；目标程序、断点信息和监控程序一般放在相应的存储器中。

仿真器一般有单步式、断点式和全速式等几种不同的执行方式。在仿真系统中，相应的执行方式是由用户在上位机中设置，这些控制信息同待装载的目标程序一起被传送给下位机。数据传送完成后，下位机等待用户的执行命令并在收到命令后开始执行。如果是单步式执行方式，下位机的控制芯片和控制电路在目标芯片每执行一条指令后都将目标芯片挂起，并将下位机存储器中的信息和目标芯片寄存器信息反馈给上位机并加以显示。上位机等待用户发出继续执行的命令并通知下位机继续执行。优点是利于细致观察代码的执行，缺点是速度很慢。断点式执行方式与此类似，只是当目标芯片执行了设有程序断点的指令或是修改了设有数据断点的数据后才被挂起，并向上位机返回数据。全速式执行方式不中断地执行完整程序，只在程序执行结束后挂起目标芯片并向上位机返回数据。断点式执行方式兼具灵活性和快速性，是最常用的。典型仿真系统的结构如图 1 所示。

常规的仿真系统中检查控制断点和数据断点是由监控程序完成的。监控程序读取目标芯片总线中的内容，获得当前执行指令的程序地址，并与存储器中存储的所有控制断点的地址进行一一比较，如果符合则表明遇到控制断点；如果监控程序发现目标芯片中当前执行指令为写数据指令，则将数据地址与存储器中存储的所有数据断点的地址进行一一比较，如果符合则表明遇到数据断点。

由于检查断点由监控程序完成，因此在执行过程中必须保证目标程序与监控程序的同步性。假设目标芯片的运行频率为 f_target，监控程序进行一遍检查需要的时间为 t_monitor，则需要保证监控程序的一遍检查需要在目标芯片执行两条指令之间的时间内完成，即 t_monitor×f_target≤1。

基金项目：国家“863”计划基金资助项目(2002AA1Z1140)
作者简介：陈东晓(1976－)，男，硕士生，主研方向：多媒体系统验证平台，优化编译技术等；高 磊、梅优良，硕士
收稿日期：2006-04-30 **E-mail：**chendongxiao_ming@126.com

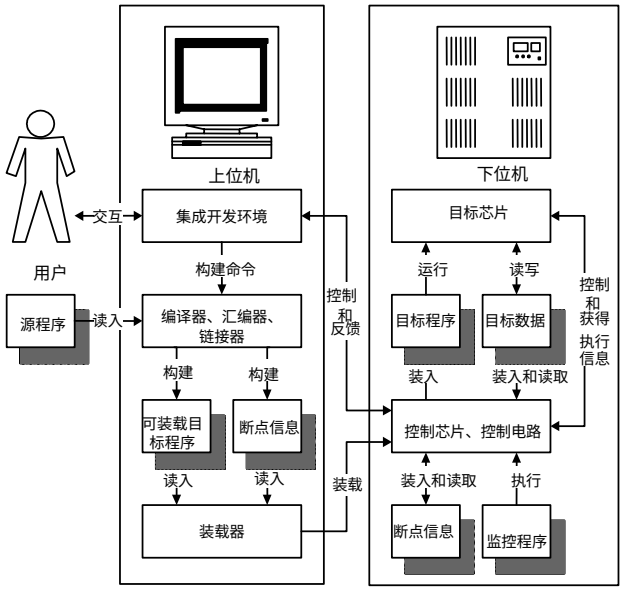

图 1 典型仿真系统的结构

为了满足这种条件，一种方法是调整目标芯片频率法。调整目标芯片的执行频率 f_target，使其小于 t_monitor 的倒数以达到目标芯片和监控程序的同步。目标芯片每执行一条指令，利用其总线变化激活监控程序进行一遍检查，检查过程耗时 t_monitor，且由于监控程序需要对存储器中的断点信息进行一一比较，该耗时随用户设置断点的数目线性增长。这时目标芯片往往只能在远低于正常的频率下执行。为了保证目标芯片的执行速度不至于太低，往往只能限制用户断点的最大数目。

另一种方法是挂起目标芯片法，利用监控程序控制目标芯片的执行。目标芯片执行一条指令后产生的总线变化激活监控程序，此时监控程序马上将目标芯片挂起并进行一遍检查。如果符合断点条件则向上位机进行反馈，否则恢复目标芯片的继续执行。这种方法不必对用户断点的数目做严格限制，而且当用户断点数目较少时，由于监控程序执行完马上恢复目标芯片的执行，不用等待固定时间，因此能达到比第 1 种方法更快的速度。但这种方法需要对目标芯片进行额外的挂起和恢复操作，增加了一定的设计复杂度，并使监控程序执行增加了一个常数时间。

在断点式执行方式下，需要检查断点来决定是否挂起目标芯片。这种检测往往非常耗时，图 2 比较了断点式和全速式执行方式在执行时的步骤。断点式执行方式下，即使没有遇到断点也需要对控制断点和数据断点进行检查，所以其执行速度明显慢于全速式执行方式。

断点式执行方式的执行步骤：

```
do{
   目标芯片执行一条指令;
   if(检查到控制断点或数据断点){
      挂起目标芯片;
      向上位机进行反馈;
      等待上位机的控制;
      if(上位机终止运行)
         break;
   }
}while(目标程序未结束);
```

全速式执行方式的执行步骤：

```
do{
   目标芯片执行一条指令;
}while(目标程序未结束);
```

针对断点执行方式下仿真速度过慢的缺点，提出一种软硬件协同的方法来提高断点执行方式下的仿真速度。

对于控制断点，利用目标程序自动对其进行检测，如果检查到断点，则通知监控程序并自动挂起目标芯片。这个过程需要通过借助汇编器和下位机的协同设计完成。如果用户在代码中某条指令上设置了控制断点，则利用汇编器在该指令后插入一组特殊的指令，首先向监控程序发出一个信号以通知监控程序执行，然后使目标芯片自动挂起。这种自动挂起又称作软挂起，指目标芯片利用指令对自身进行挂起。这样，不需要在目标芯片每执行一条指令后都调用监控程序进行控制断点检查，目标芯片执行到设有控制断点的指令后会自动通知监控程序并挂起。监控程序收到通知后向上位机进行反馈并等待上位机的进一步控制。

对于数据断点，由于无法静态(编译时)判断哪条指令会修改某个特定地址的数据，因此数据断点的检查相对困难，必须采用某种技巧来改进监控过程。利用链接器和下位机的协同设计完成这种监控过程。首先对数据存储器的空间使用进行一定的约定，划出某块特定的保留地址专门放置被用户设置为断点的数据(或下文提到的其它必须在断点数据周围的数据)，该保留地址的特征可以作为启动监控程序检查数据断点的条件。我们称该保留地址为数据断点保留地址。在我们的设计中数据断点可以加在各种全局变量上，该全局变量可能是一个独立的变量，也可能是某个聚合体(aggregation，如 C 语言中的 struct、union，C++中的 class 等)中的一个成员变量，或者是某个数组的一个元素。控制所有全局数据的地址绑定在链接阶段完成。对于独立的全局变量，如果被设置为数据断点，则可以利用链接器将该变量地址绑定到数据断点保留地址处；对于聚合体中的成员或数组中的元素，由于语义上要求该变量和其所在的聚合体或数组中的其它变量在空间上的连续性，需要将整个聚合体或数组绑定到保留地址处。这样做的结果是所有被设置断点的数据都被分配到了数据断点保留地址处，但保留地址处的数据并不一定都是设置了断点的数据。下位机中，目标芯片每执行一条写某个特定地址的指令后，只需要用一组非常简单的电路(可以认为其执行是实时的)判断该地址是否属于数据断点保留地址，如果不是，则不做任何动作，目标程序继续执行；如果是，则通知启动监控程序检查该数据是否确实被设置为断点并挂起芯片。由于目标芯片首先被挂起，此时监控程序可以对被写入数据的地址和存储器中的数据断点地址进行一一比较来检查该数据是否的确被设为断点。如果检查结果为真，则向上位机进行反馈，否则，恢复目标芯片的运行。

由于一般来说，仿真时绝大多数数据并不会被设置为数据断点，即不会被分配到数据断点保留地址处，因此采用这种方法，绝大多数写地址指令并不会引发比较耗时的监控程序的执行。

2 快速断点仿真器的设计方案

根据上述设计原理，提出了一个快速断点仿真器的设计方案，并进行仿真和部分实现。该方案以MXIC公司的MX96037 芯片[2]为目标芯片，采用Cypress公司的CY7C68013[3]作为控制芯片，利用FPGA设计控制电路。在上位机中提供集成了编辑器、汇编器、链接器和装载器[4]的集成开发环境。上位机和下位机采用并口和USB接口相连。

图 2 是我们开发的集成开发环境的主界面，该集成开发环境包括工程管理系统和编辑器，并能调用专门为快速断点仿真器设计的汇编器和链接器。

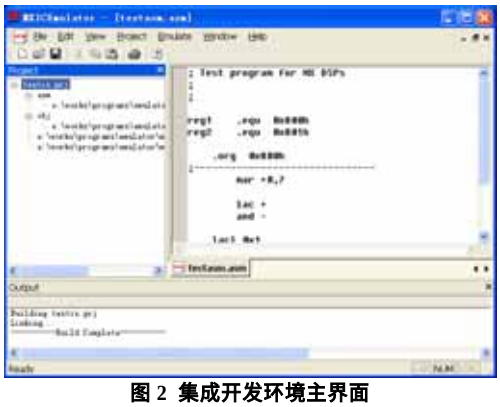

图 2 集成开发环境主界面

一般的汇编器由扫描和代码产生两部分组成。在实现上，汇编器的扫描一般分为两遍[1]。第 1 遍扫描记录汇编程序中的符号，产生一个符号表，并分析伪汇编指令；第 2 遍扫描根据符号表的内容，在每读入一条汇编语句后进行模板匹配并输出目标代码。对汇编器的扫描过程进行修改，在第 1 遍扫描过程中添加记录断点的功能，记录用户设置的所有控制断点和数据断点并将数据断点的记录导出到一个数据断点描述文件中；第 2 遍扫描中，每输出一次目标代码后汇编器调用一个回调函数，利用第 1 遍扫描的结果判断该语句是否被设置了控制断点，如果是，则在其后添加一小段控制代码，完成挂起目标芯片并通知监控程序的功能。对于MX96037 芯片，添加的控制代码为

```
SAH 0x1b    #1
SAL 0xff    #2
//1、2 两句代码对目标地址 0x1bff 发出一个写命令。
Out 0x0400 0x03      #3
//DSP 对自己发出一个软中断命令。
```

其中第 1、第 2 两句代码通过改写一个约定好的保留的内存地址(该保留地址不同于数据断点保留地址)，在目标芯片的数据引脚上发出一个特定信号，改写的内容并不重要，监控程序通过检查数据线发现这个改写信号后将被激活。该保留地址在我们的设计中是 0x1bff，链接器将保证该保留地址不做其它用途。目标芯片执行第 3 句代码对自己进行软挂起，在收到一个外部中断后，目标芯片结束挂起状态。该外部中断由 FPGA 控制电路在确定需要继续执行时发出。

在我们的实现中，对数据的内存绑定都留在链接阶段完成。在链接的过程中，需要读取数据断点的内容并确定哪些数据需要绑定到断点数据保留地址处，在分配过程中，要将这些数据绑定到断点数据保留地址并避免将其它数据绑定到该保留地址。在我们的设计中，约定保留地址为 0x1c00 到 0x1fff 的范围。

图 3 是下位机的结构示意图。其中需要特别提到的是图 4 中阴影所示区域，该区域内容在物理上实际为 FPGA 的一部分，但在逻辑上为一个单独的功能单元，为了方便讨论，将其单独划出。由于约定的数据断点保留地址为 0x1c00~0x1fff 的范围，即数据地址的高 5 位全为 1 的地址，因此可以对目标芯片地址总线高 5 位进行“与”操作，操作的结果为真即表示写操作的地址在数据断点保留区域内。利用该结果作为电平信号加在目标芯片的 hold 管脚即可达到挂起目标芯片的效果，同时该结果还将传递给 FPGA 的其它部分以通知监控程序开始执行。

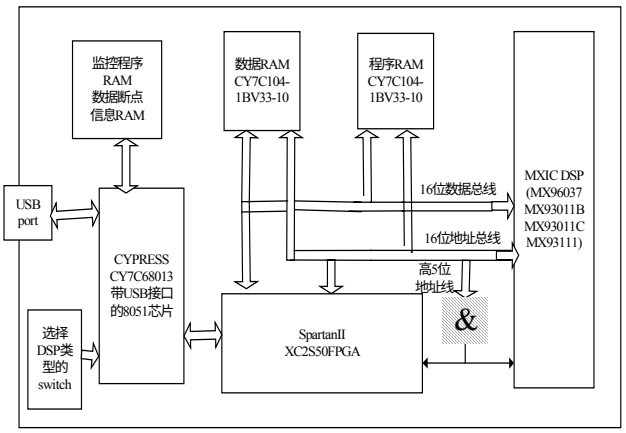

图 3 下位机结构示意图

该设计中默认目标芯片为 MX93011，该芯片的工作频率为 32.768kHz 到 32.256MHz。在我们的设计中该芯片工作在 20MHz，8051 的工作频率为 48MHz，FPGA 的工作频率为 100MHz，数据断点信息 RAM 可容纳最多 4k 个数据断点。

3 仿真结果和结论

我们部分实现了快速断点仿真器。为了方便比较，给出了该方案与第 1 部分介绍的调整目标芯片频率法和挂起目标芯片法的仿真比较结果。由于断点调试过程需要人的操作和参与，操作人员在上位机的操作时间往往占断点仿真过程的多数时间，且实际程序中加入断点的数目严重影响仿真器执行速度，因此主要比较仿真器三方面的性能：控制断点的反应时间(注：本文所有反应时间为从目标芯片开始执行某条断点指令到准备向上位机进行反馈之间的时间)，数据断点的反应时间和在未遇到断点时仿真器的执行速度。其中第 3 点决定了断点仿真器的执行速度，是我们关心的主要内容。

表 1 几种方法的仿真结果比较

性能指标	控制/数据断点数量	调整目标芯片法	挂起目标芯片法	快速断点仿真器
控制断点反应时间(单位 ns)	控制断点 1	3 010	90	130
	控制断点 10	3 100	360	130
	控制断点 100	4 000	3 060	130
数据断点反应时间(单位 ns)	数据断点 1	3 010	90	30
	数据断点 10	3 100	360	310
	数据断点 100	4 000	3 060	3 010
未遇到断点时仿真器执行速度(单位 mips)	各 1	0.15	5.56	19.99
	各 10	0.15	1.38	19.64
	各 100	0.15	0.16	7.24

表 1 列出了在分别设置 1、10、100 个控制断点和数据断点条件下的仿真结果。测试程序是对一个 256×256 大小的 24 位位图进行 8 点分块 DCT 和 IDCT 的程序，旨在反映 DSP 常见用途下的仿真结果。其中调整目标芯片频率法中可容纳控制断点和数据断点的上限都设定为各 100 个。调整目标芯片频率法和挂起目标芯片法的某些系统参数(如控制断点数据搜索速度等)利用快速断点仿真器的参数估计得来。对于随具体断点位置不同而可变的性能指标，给出平均结果。调整目标芯片法的控制断点反应时间和数据断点反应时间都为较大的常数，其执行速度较慢。挂起目标芯片法的反应时间随断点数量增加而线性增长，未遇到断点时的执行速度随断点增加而下降。快速断点仿真器能够获得较小的控制断点反应

（下转第 251 页）