RISC-V架构的HPE和VTF中断技术研究

原创 杨勇 嵌入式系统专家之声 2025年10月29日 08:02 北京

2025.10.29

本文字数：3759
预计阅读时长：10分钟

摘要

本文介绍了一款基于RISC-V架构的快速可编程中断控制器（FPIC），相比较标准RISC-V中断控制器PLIC，FPIC具备硬件压栈（HPE）和免表（VTF）中断技术。对比沁恒自研RISC-V微处理器青稞V4的中断控制器FPIC与PLIC中断处理流程的差异，并以通用32位互联型MCU赤菟V307为实验芯片，验证开启与不开启HPE、VTF模式时的中断延迟实测差异。

关键词

RISC-V；FPIC；HPE；VTF；青稞V4；赤菟307

作者：杨勇

中图分类号：TP31

文献标识码：A

0 引言

随着计算机科学技术的快速发展，嵌入式应用领域对数据传输速度、运行功耗等提出了越来越高的要求。其中物联网是集成电路未来十年的主战场，机器学习的加入使得AIoT技术增加了对MCU的高性能需求，其中最为关键的技术指标是快速中断响应。

当前面向AIoT应用的MCU绝大多数基于ARM指令集架构，其存在昂贵的商业授权和严格的专利壁垒限制。开放指令集RISC-V的兴起为我国突破自主可控处理器内核关键技术提供了前所未有的机遇。

标准RISC-V中断控制器PLIC不能达到AIoT 对中断响应实时性的要求。因为PLIC是一种集中式管理的中断控制系统，每种特权模式提供单独的中断信号给内核。而嵌入式的MCU一般不会支持所有的特权模式，有的可能只有机器模式和用户模式，还有一些可能只支持机器模式，在单个模式的MCU应用中如果采用PLIC中断控制器就无法做到中断抢占功能。

针对标准RISC-V PLIC中断控制器的弊端，沁恒微电子自研设计了快速可编程中断控制器（FPIC），包含了硬件压栈（HPE）、免表（VTF）中断技术。

1 中断响应流程

标准RISC-V中断控制器PLIC的中断响应流程如图1所示。通常，RISC-V的中断控制器PLIC，在得到中断响应请求时会停止当前程序流程并自动保存或修改PC值，由软件方式确定中断源后跳转到对应的中断服务函数中。而大部分MCU的中断来源于外部中断，外部中断的种类多、发生时间不确定，所以系统在编译过程中是无法知道调用中断服务函数时的上下文内容。因此，编译软件会在进中断服务函数前将Caller-save类型通用寄存器数据进行压栈保存，离开中断服务函数时再出栈恢复，即将之前保存的数据再写回到通用寄存器中。

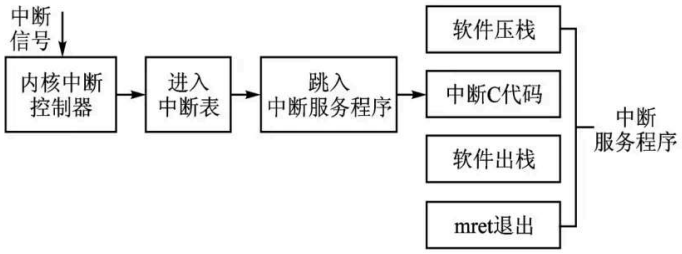

图1 标准**RISC-V PLIC**中断响应流程

沁恒自研快速可编程中断控制器FPIC的中断响应流程如图2所示。与PLIC相比，FPIC减少了中断向量表查表动作、免去软件压栈操作。理论上，在标量处理器上软件压栈至少需要17个周期，再加上取指时间、延迟等，往往大于19个周期。在一些低成本嵌入式产品上，软件压栈至少需要几十个周期，这在对中断实时性要求较高的AIoT应用场景中是无法接受的。

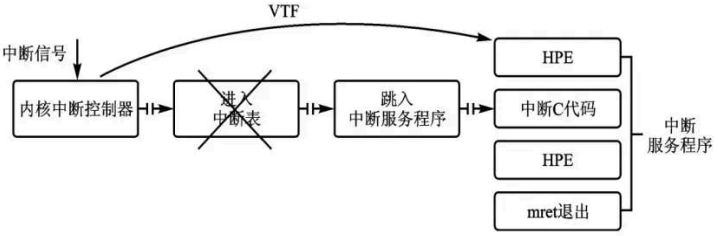

图2 沁恒自研**RISC-V FPIC**中断响应流程

2 中断控制器

标准RISC-V中断控制器如图3所示。MCU应用中一个外设的数据接收完成中断，一般都会设置其为较高优先级且可以打断其他中断，以确保其数据接收的实时性，而PLIC中断控制器做不到单个模式的中断抢占，所以达不到AIoT应用中对中断响应实时性的需求。另外，MCU的中断源个数从几个到几十个不等，而PLIC采用统一入口管理模式，会进一步增加中断判断的时间，从而进一步增加了中断响应延迟。

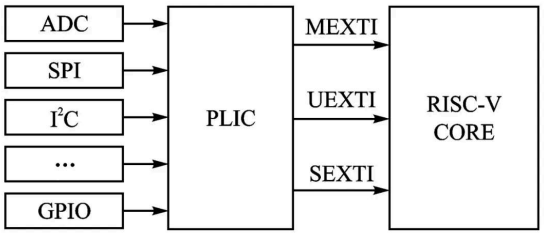

图3 **RISC-V PLIC**中断控制器

沁恒自研RISC-V快速可编程中断控制器如图4所示。在沁恒FPIC中断控制器中，外设各中断分别直接送入内核，内核根据送入的中断类型进行中断函数跳转。其中中断跳转设计了两种方式：一是快速中断技术，通过查询中断向量表进行中断函数寻址，不再通过统一的入口管理和进一步判断，减少了中断响应延时（其中，中断向量表可以存放跳转指令和绝对地址，存放绝对地址时可以规避JAL±1M字节寻址范围的限制）。二是VTF免表中断技术，即发生中断时不用通过查询中断向量表即可直接完成中断函数的寻址，可以进一步减少中断的响应延迟。

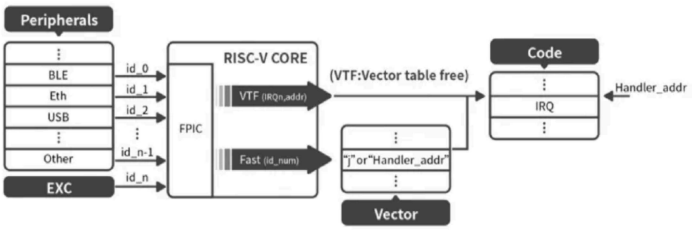

图4 沁恒自研**RISC-V FPIC**中断控制器

3 中断延时

采用沁恒通用32位互联型MCU赤菟V307做中断延迟对比实测实验。赤菟V307的微处理器为青稞V4，其中Load和Store均为单周期指令。

实验方法：采用Systick计数中断，计算值从零开始，计数比较值设置为CNT1，本例取值0x20。计数到CNT1后进中断且计数器不停止，在中断函数中再次读取计数器值CNT2，两条读计数器值的汇编代码占用时间为Tc，中断延迟时间Ts=CNT2-CNT1-Tc。虽然此方法可能受到内部的预取、流水线等影响，但最终得到的中断延迟时间数据仍具有普遍代表性。

3.1 软件压栈

在C语言代码编译环境下，使用"__attribute__((interrupt))"声明，目标文件的中断函数开头和结尾添加压栈出栈指令。基于赤菟V307单片机的实验中，不开启HPE和VTF，即采用软件压栈的常规快速中断模式。常规软件压栈实测代码反汇编如图5所示。

进中断后读取计数值CNT2=0x3F,Tc=2，软件压栈存在跨区访问不同存储器带来的额外2个周期数，所以最终Ts=27。从图5中的汇编代码可以看出，软件压栈额外增加了17条指令时间。

```
30c:   20078793          addi    a5,a5,512 # 200 <IRQ_Handler>
310:   8ff5                 and a5,a5,a3
312:   0017e793          ori a5,a5,1
316:   d33c                 sw  a5,96(a4)
318:   e000f7b7          lui a5,0xe000f
31c:   0007a423          sw  zero,8(a5) # e000f008 <_eusrstack+0xc0007008>
320:   0007a623          sw  zero,12(a5)              设置定时器计数初始值为0
324:   02000713          li  a4,32
328:   cb98                 sw  a4,16(a5)            设置比较值为0x20
32a:   0007a623          sw  zero,12(a5)
32e:   471d                 li  a4,7                 使能定时器
330:   c398                 sw  a4,0(a5)
332:   0001                 nop
334:   0001                 nop

00000200 <IRQ_Handler>:
200:   7139                 addi    sp,sp,-64
202:   d62a                 sw  a0,44(sp)
204:   d42e                 sw  a1,40(sp)
206:   cc3e                 sw  a5,24(sp)
208:   de06                 sw  ra,60(sp)
20a:   e000f7b7          lui a5,0xe000f
20e:   dc16                 sw  t0,56(sp)
210:   da1a                 sw  t1,52(sp)
212:   d81e                 sw  t2,48(sp)
214:   d232                 sw  a2,36(sp)
216:   d036                 sw  a3,32(sp)
218:   ce3a                 sw  a4,28(sp)
21a:   ca42                 sw  a6,20(sp)
21c:   c846                 sw  a7,16(sp)
21e:   c672                 sw  t3,12(sp)
220:   c476                 sw  t4,8(sp)
222:   c27a                 sw  t5,4(sp)
224:   c07e                 sw  t6,0(sp)
226:   0007a023          sw  zero,0(a5) # e000f000 <_eusrstack+0xc0007000>
22a:   478c                 lw  a1,8(a5)             关闭定时器
22c:   00006537          lui a0,0x6
230:   9e050513          addi    a0,a0,-1568 # 59e0 <_sbrk+0x60>
234:   80b1a823          sw  a1,-2032(gp) # 200001f0 <_edata>
238:   456010ef          jal ra,168e <iprintf>
23c:   a001                 j   23c <IRQ_Handler+0x3c>
```

图**5** 软件压栈反汇编代码

3.2 硬件压栈（HPE）

在MRS集成开发环境中使用使用"__attribute__((interrupt("WCH-Interrupt-fast")))"声明，通过代码配置开启硬件压栈　HPE功能。硬件压栈反汇编代码如图6所示。

```
2d0:   e000e7b7          lui a5,0xe000e
2d4:   6705                 lui a4,0x1
2d6:   10e7a023          sw  a4,256(a5) # e000e100 <_eusrstack+0xc0006100>
2da:   e000f7b7          lui a5,0xe000f
2de:   0007a423          sw  zero,8(a5) # e000f008 <_eusrstack+0xc0007008>
2e2:   0007a623          sw  zero,12(a5)              设置定时器计数初始值为0
2e6:   02000713          li  a4,32
2ea:   cb98                 sw  a4,16(a5)
2ec:   0007a623          sw  zero,12(a5)              设置比较值为0x20
2f0:   471d                 li  a4,7
2f2:   c398                 sw  a4,0(a5)             使能定时器
2f4:   0001                 nop
2f6:   0001                 nop
2f8:   0001                 nop

00000200 <IRQ_Handler>:
200:   e000f7b7          lui a5,0xe000f
204:   478c                 lw  a1,8(a5)             读计数器值
206:   00006537          lui a0,0x6
20a:   9c450513          addi    a0,a0,-1596 # 59c4 <_sbrk+0x62>
20e:   80b1a823          sw  a1,-2032(gp) # 200001f0 <_edata>
212:   45c010ef          jal ra,166e <iprintf>
216:   a001                 j   216 <IRQ_Handler+0x16>
```

图**6** 硬件压栈反汇编代码

进中断后读取计数值CNT2=0x2C,Tc=2，所以最终Ts=10。仅开启HPE时的中断效率是软件压栈模式时的2.7倍，如图7所示。

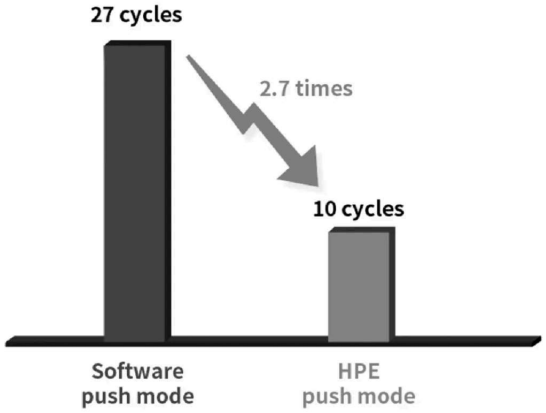

图7 硬件压栈与软件压栈中断效率测试对比

3.3 VTF免表中断技术

发生中断时，不用通过查询中断向量表即可直接完成中断函数的寻址，可以进一步减少中断的响应延迟。同时开启HPE+VTF，测试中断响应延时。HPE+VTF中断反汇编代码如图8所示。

```
2ea:   20078793          addi    a5,a5,512 # 200 <IRQ_Handler>   设置VTF地址
2ee:   8ff5                  and a5,a5,a3
2f0:   0017e793          ori a5,a5,1
2f4:   d33c                  sw  a5,96(a4)                       使能VTF
2f6:   e000f7b7          lui a5,0xe000f
2fa:   0007a423          sw  zero,8(a5) # e000f008 <_eusrstack+0xc0007008>
2fe:   0007a623          sw  zero,12(a5)                         设置定时器计数初始值为0
302:   02000713          li  a4,32
306:   cb98                  sw  a4,16(a5)                       设置定时器比较值为0x20
308:   0007a623          sw  zero,12(a5)
30c:   471d                  li  a4,7
30e:   c398                  sw  a4,0(a5)                        使能定时器
310:   0001                  nop
312:   0001                  nop
314:   0001                  nop
316:   0001                  nop
318:   0001                  nop

00000200 <IRQ_Handler>:
    200:   e000f7b7          lui a5,0xe000f
    204:   478c                  lw  a1,8(a5)                    读定时器计数值
    206:   00006537          lui a0,0x6
    20a:   9e050513          addi    a0,a0,-1568 # 59e0 <_sbrk+0x62>
    20e:   80b1a823          sw  a1,-2032(gp) # 200001f0 <_edata>
    212:   478010ef          jal ra,168a <iprintf>
    216:   a001                  j   216 <IRQ_Handler+0x16>
```

图8 HPE+VTF中断反汇编代码

进中断后，读取计数值CNT2=0x2A,Tc=2，所以最终Ts=8。开启HPE+VTF时的中断效率是软件压栈模式时的3.4倍，相较于只开启HPE，VTF减少了2个周期的查表时间，如图9所示。目前主流的MCU大多数为非零等待Flash MCU，如果采用VTF技术，理论上可以节约5个周期左右的时间。本文实验所采用的CH32V307的Flash为零等待。

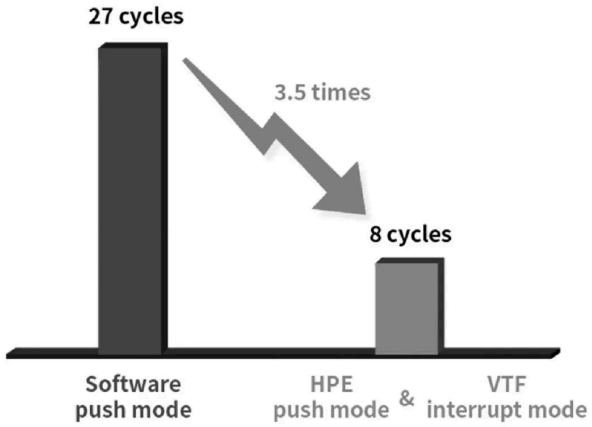

图9 HPE+VTF与软件压栈中断效率实测对比

4 结语

标准RISC-V中断控制器PLIC是一种集中式管理的中断控制系统，每种特权模式提供单独的中断信号给内核。在单个模式的嵌入式MCU应用中，如果采用PLIC中断控制器就没法做到中断抢占功能。另外，一个嵌入式MCU的中断源个数从几个到几十个不等，如果采用PLIC统一入口管理方式，会增加进一步判断的时间，从而增加中断响应延时。

在一些对实时性和功耗要求较高的场合，都会要求将代码放到RAM中运行。但PLIC会受限于JAL 跳转指令±1MB的寻址范围限制，中断处理函数和中断向量表需要放到同一块存储区域。如果都放到RAM中，那么所有的中断处理函数都需要一起放到　RAM中，这样RAM资源将严重受限。

标准RISC-V中断控制器采用软件压栈，一个标量RISC-V处理器至少会额外增加17个周期数，此外在中断服务函数和软件堆栈空间处于同一块RAM的情况下，还会造成取指令、压栈／出栈冲突的情况，进一步降低函数执行效率。

采用FPIC中断控制器，一是可以避免集中式管理的耗时方式，同时可实现单个模式下的中断抢占功能；二是中断跳转向量表可以存放绝对地址，解决了JAL寻址范围受限问题；三是采用硬件压栈，可显著提高中断效率；四是VTF免表技术可以进一步提升中断响应效率。

参考文献

[1]凌明，王学香，单伟伟．嵌入式系统——从SoC芯片到系统[M]．北京：电子工业出版社，2017.

[2]王宜怀．嵌入式计数基础与实践[M]．北京：清华大学出版社，2007.

[3]蔡亮．RISC-V架构的高速接口芯片设计及USB3.0应用[J]．单片机与嵌入式系统应用，2020(11).

[4]何小庆．RISC-V处理器嵌入式开发概述[J]．单片机与嵌入式系统应用，2020(11).

[5]Karyofyllis Patsidis, Dimitris Konstantinou, Chrysostomos Nico-poulos, et al. A low-cost synthesizable RISC- V dual-issue pro-cessor core leveraging the compressed Instruction Set Extension [J]. Microprocessors and Microsystems,2018(61):1-10.

（作者单位：南京沁恒微电子股份有限公司，南京 210012）

（本文由《单片机与嵌入式系统应用》杂志社授权发表，原文刊发在2022年第2期）

END

期刊论文 · 目录

上一篇 · 高实时要求的嵌入式智能计算云平台设计