RISC-V 的数据流 T*指令支持 小组
毕业设计任务分解
2020-10-31

任务描述：

为 RISC-V 提供数据流任务调度的的硬件支持，加快数据流任务就绪条件的的检测速度。该硬件工作原理如下：

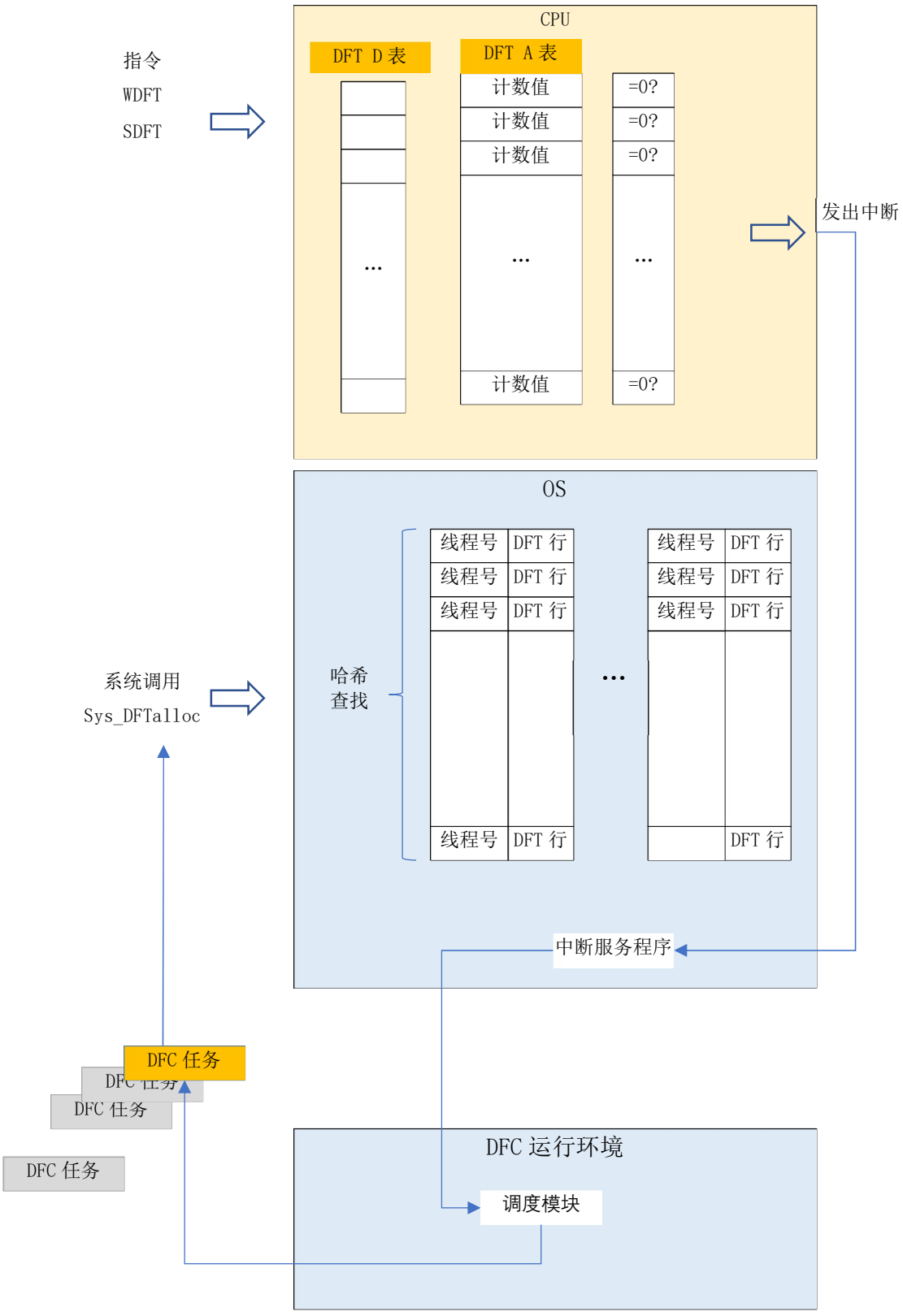
- ✧ RISC-V 处理器内部增设一个“输入条件”硬件表 DFT，表长 L=1024，宽度 W=32bit，用于记录数据流任务的“未准备好”数据项的个数。
- ✧ 该表写入必须是在系统态/内核态，由内核代码负责写入初值。内核代码需要维护一个软件表，用于记录数据流任务的线程号和在硬件表中的位置。使用 WDFT $R1,$$2 指令，其中 R1 指出表的行号，R2 给出写入的数值；
- ✧ 数据流任务向发出系统调用，向操作系统请求分配一个 DFT 行。
- ✧ 数据流任务在用户态发出减一操作指令，SDFT $R1，将 R1 指出的行的计数值-1。
- ✧ 当计数值=0 时，发出指定中断。

软件触发调度的问题：

1） 每次需要“互斥”地访问计数值，以便正确地修改和判定是否为 0，从而才能触发一下级任务；
2） 使用软件线程，无法高效实现“硬件”切换。

硬件优势：

1） 当计算量极小时，每一个互斥访问，都需要经过系统调用访问信号量（比小计算量任务还要耗时很多），使用硬件计数后，只需要一次系统调用；可以分析出，任务的大小和系统调用的开销，双方使得硬件收益的平衡点，作为是否应用该技术的评价依据。
2） 任务激活如果采用“线程”使得调度开销过高，如果使用硬件将任务“链接”起来，可以加快调度过程——前提是任务必须是 no side-effect 的（除了出口以外，只访问堆栈内部数据）。可以分析出，硬件链接执行方式，比软件线程方式快多少。


CPU
指令
WDFT
SDFT
DFT D 表
DFT A 表
计数值
=0?
发出中断
OS
线程号
DFT 行
哈希
查找
系统调用
Sys_DFTalloc
中断服务程序
DFC 任务
DFC 运行环境
调度模块

任务分解及进度计划：

✧ 1 T*指令集接口实现==========杜

阅读 RISC-V 的的指令扩展规范，确定指令编码； （含 chisel 环境熟悉）

分析 RISC-V 的 mini 或 sodor 实现，修改器指令译码电路发出合适的控制电路，与 DFT 硬件表电路相连接

设计指令集的 test 完成处理器级的仿真，验证指令“接口”功能正常

GCC 中 C 语言的内嵌汇编部分，增加 T*指令的实现

✧ 2 T*指令集功能实现==========谢

设计 DFT 硬件表完成功能实现 （含 chisel 以及开发环境熟悉）

设计 DFT 硬件的 tester 完成仿真验证

GCC 中汇编器的修改，增加 T*指令

✧ 3 T*指令集的 T*库以及测试验证软件==========黄

由于在硬件未准备好之前开始设计，毕设任务可以将上述函数内部实现为“纯软件”模拟的方式——模拟计数功能和发出回调函数，开发环境为 x86+Linux+DFC。

熟悉 DFC

将 DFC 环境中的数据流任务线程修改，被调度运行前（前一级任务准备就绪后）需要向操作系统注册，获得 DFT 表的行号、写入所需的输入数据计数值。

由于 T*指令不被 GCC 所支持，因此需要编写机器码函数——方法是编写一个替身函数然后修改里面的二进制代码，使得其指令编码符合 T*指令。（这个需要在 RISC-V 仿真环境验证）

编写中断服务程序，用于将上述任务标志成就绪任务。

工作原理描述：

1）DFC runtime 中，数据流任务的主动数据个数纪录载在 XXXXXXX，主动数据的地址纪录在 XXXXXX。

2）当一个数据流任务就绪后，其直接后集结点将成为“后续结点”，放入硬件标中等待激活。每个任务需要占用 A 表 1 项，每个主动数据需要占用 B 表 1 项。

3）操作系统维护着 AB 表的使用纪录，负责分配和回收。

4）当 runtime 为一个“后续结点”申请 AB 表资源的时候，发出一个系统调用，返回 AB 标中的空槽位置。

5）当 A 表的某一项计数值降为 0，发出一个中断（异常），并在特定寄存器中给出行号或线程号，标明下一个任务可以被调度运行，当前仍有软件进行调度——本意是通过硬件激活而执行——使用线程部切换的形式完成数据流任务的切换。