# 中断

## CSR

class Priv extends Bundle {

  val m = Output(Bool())

  val h = Output(Bool())

  val s = Output(Bool())

  val u = Output(Bool())

}

class Interrupt extends Bundle {

  val e = new Priv()

  val t = new Priv()

  val s = new Priv()

}

CSR中定义了mideleg寄存器

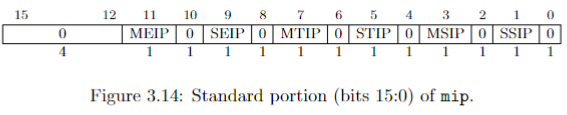
  val mideleg    = RegInit(UInt(XLEN.W), 0.U) // 中断代理寄存器

CSR中mip的值等于mip寄存器的值 或上 当前外部引起的中断的值，因为中断可以由外部引起也可以由软件写入mip寄存器引起，外部引起的中断由mipWire的值表示，软件使用CSRW等指令写入mip寄存器由mipReg的值表示，最终的mip表现成两者值的或的样子

  val mipWire    = WireInit(0.U.asTypeOf(new Interrupt))

  val mipReg     = RegInit(UInt(XLEN.W), 0.U)

  val mip        = mipWire.asUInt | mipReg // 中断挂起寄存器



其中mipWire当前值的定义如下

// interrupts

  val mtip = io.ext\_int.ti

  val meip = io.ext\_int.ei

  val msip = io.ext\_int.si

  mipWire.t.m := mtip

  mipWire.e.m := meip

  mipWire.s.m := msip

当前是否存在中断与mip的值有关

  val mip\_raise\_interrupt = WireInit(mip.asTypeOf(new Interrupt()))

使用下述语句使s模式可以接收到外部中断

  val seip                = meip

  mip\_raise\_interrupt.e.s := mip.asTypeOf(new Interrupt).e.s | seip

下面的mstatusBundle就是mstatus，只是转换了下格式，ideleg的值是mideleg和当前中断值的与，ideleg第i位的值为1则表示发生了第i类中断并且由S模式处理，否则由M模式处理。priviledgedEnableDetect函数由于查询当前权限是否允许发生该类中断。

  val mstatusBundle = mstatus.asTypeOf(new Mstatus())

  val ideleg = (mideleg & mip\_raise\_interrupt.asUInt)

  def priviledgedEnableDetect(x: Bool): Bool = Mux(

    x,

    ((mode === ModeS) && mstatusBundle.ie.s) || (mode < ModeS),

    ((mode === ModeM) && mstatusBundle.ie.m) || (mode < ModeM)

  )

  val interrupt\_enable = Wire(Vec(INT\_WID, Bool()))

下面的函数的y的值对应了ideleg的每一位，调用priviledgedEnableDetect函数，可以得到最终的中断使能向量

  interrupt\_enable.zip(ideleg.asBools).map { case (x, y) => x := priviledgedEnableDetect(y) }

如果以下所有条件**均为真**，则中断i将捕获到M模式（导致特权模式更改为M模式）：（a）当前特权模式为M并且设置了mstatus寄存器中的MIE位，或者当前特权模式的特权低于M模式；（b）在mip和mie中都设置了位i；（c）如果寄存器mideleg存在，则在mideleg中没有设置位i，这段话来自手册定义

mie与上mip\_raise\_interrupt对应了条件b，interrupt\_enable对应了条件a和c；当条件满足时，中断会被标记到当前正在译码单元中执行的指令上，当该指令运行到访存单元时，会引发中断，进入中断处理程序

  io.decodeUnit.interrupt := mie(INT\_WID - 1, 0) & mip\_raise\_interrupt.asUInt & interrupt\_enable.asUInt

## Decoder Unit

下面的代码在译码单元中，将中断标记到指令上

(0 until (INT\_WID)).foreach(j => io.executeStage.inst(i).ex.interrupt(j) := io.csr.interrupt(j))