在大核里

---

stride load

RPT ：(pc : 40 addr : 40 stride : 16 state : 2 最内层检测位 : 1)

当发现检测位为1，可以去清空VTT和FLR ，重新开始

每当重新遇到启动discoder mode 的pc时， 会清空所有的检测位

Taint Tracker

VTT ： 依赖链的建立 ,污点向量记录

FLR : 记录最后一个load ，更新FLR时，清空SBB和LCR

loop bound detector

SBB:是否处理过分支

LCR:保存上一个比较指令的源和目标寄存器的ID (x86是这样的) ， riscv的话，累加指令？

bne s0,s4 ,1031a

beqz s2,10362

小核的执行单元和访存单元 ，写回单元（） ， 需不需要写小核的寄存器 ？

发一条小核执行一条 ？

什么时候发送?

找到 stride load 和 loop bound , 再次进入前一个stride load 的时候

PCv 来记录 stride load pc ，在发现检测位为1时可以去更新PCv, 并且清空VTT和 FLR

下一轮会把所有标记为脏的指令全部都发送到一个结构里去 ， 然后再用一个结构向量化转发给多个小核（可以在标记的时候就发送吗 ？）

发送给某个结构的内容？

能使得小核成功执行指令的 部分

stride load addr step ， 可能不止一条stride load ,

只有stride load 需要电路控制 step

一些算数运算 rs1 rs2 rdata1 rdata2 op , 要放寄存器序号需要重命名， 可以不放寄存器号，可以直接映射到id去 （1）in-order核加一部分寄存器 （2）小核维护一个freeReg 表 ， 完成之后需要恢复

jar ?

depend load rs1 imm

boom Uop -> rocket decode , 大部分可以去除

提取 ？

dispatch -> uop

中间数据都需要通过 exe 读取 ， 最快的应该是从 iregister_read的时候读取，

需要提取的中间数据 ： 没有被VTT标记的寄存器值

```
// rocket decode
def default: List[BitPat] =
                //           jal
                        renf1               fence.i
                //   val     | jalr
                          | renf2             |
                //   | fp_val| | renx2
                           | | renf3           |
                //   | | rocc| | | renx1               s_alu1
          mem_val       | | | wfd           |
                //   | | | br| | | | scie      s_alu2  |       imm    dw     alu
           | mem_cmd     | | | | mul         |
                //   | | | | | | | | | zbk     |       |       |      |      |
           | |           | | | | | div       | fence
                //   | | | | | | | | | | zkn   |       |       |      |      |
           | |           | | | | | | wxd     | | amo
                //   | | | | | | | | | | | zks |       |       |      |      |
           | |           | | | | | | | |       | | | dp
                List(N,X,X,X,X,X,X,X,X,X,X,X,  A2_X,   A1_X,   IMM_X, DW_X,
 aluFn.FN_X,   N,M_X,        X,X,X,X,X,X,X,CSR.X,X,X,X,X)
```

```
//                                                                  frs3_en
                 wakeup_delay
            //     is val inst?                                                  |
imm sel                     |    bypassable (aka, known/fixed latency)
            //     |  is fp inst?                                               |
|     uses_ldq              |    |  is_br
            //     |  |  is single-prec?                       rs1 regtype     |
|     |  uses_stq           |    |  |
            //     |  |  |  micro-code                          |       rs2 type|
|     |  |  is_amo          |    |  |
            //     |  |  |  |         iq-type  func unit        |       |       |
|     |  |  |  is_fence     |    |  |
            //     |  |  |  |         |        |                |       |       |
|     |  |  |  |  is_fencei |    |  |  is breakpoint or ecall?
            //     |  |  |  |         |        |        dst     |       |       |
|     |  |  |  |  |  mem    |    |  |  |  is unique? (clear pipeline for it)
            //     |  |  |  |         |        |        regtype |       |       |
|     |  |  |  |  |  cmd    |    |  |  |  |  flush on commit
            //     |  |  |  |         |        |        |       |       |       |
|     |  |  |  |  |  |      |    |  |  |  |  |  csr cmd
            //     |  |  |  |         |        |        |       |       |       |
|     |  |  |  |  |  |      |    |  |  |  |  |  |
              List(N, N, X, uopX    , IQT_INT, FU_X   , RT_X  , DC2    ,DC2
,X, IS_X, X, X, X, X, N, M_X,   DC2, X, X, N, N, X, CSR.X)
```

附录

参考文献