riscv-trace-spec

problem 1

p15

`This works by tracking execution from a known start address and sending messages about the address deltas taken by the program`

从已知的地址追踪执行是指程序第一条指令的地址？还是任何一个给定的能够跑到的地址？

什么叫做程序获取的地址增量消息？谁发送？

problem 2

p15

`in most implementations, it will supply a large amount of data (instruction address, instruction type, context information, ...) for each core execution clock cycle`

for each core execution clock cycle是什么意思？

problem 3

p16

`Because of this, there is no need to report sequential instructions in the trace, only whether the branches were taken or not and the address of taken indirect branches or jumps. If the program counter is changed by an amount that cannot be determined from the execution binary, the trace decoder needs to be given the destination address (i.e. the address of the next valid instruction). Examples of this are indirect branches or jumps, where the next instruction address is determined by the contents of a register rather than a constant embedded in the program binary.`

- 顺序指令不用trace
- 分支指令是否跳转

  ?

  bne

  beq

  bge

  bgeu

  blt

  bltuf

  jal
- 间接分支/跳转的地址，以及一切不能从二进制文件得到（可能从寄存器得到）的PC增量的下一条有效地址

?

jalr

ecall

ebreak

sret

mret

异常

中断

problem 4

p16

`The decoder generally does not know where an interrupt occurs in the instruction sequence, so the trace encoder must report the address where normal program flow ceased, as well as give an indication of the asynchronous destination which may be as simple as reporting the exception type. When an interrupt or exception occurs, or the processor is halted, the final instruction retired beforehand must be included in the trace.`

需要trace三个东西：异常发生前最后一条指令、异常类型、异常后第一条指令

problem 5

p19

`This chapter describes the fields required to control the Trace Encoder. How fields are organized and accessed (e.g packet based or memory mapped) is outside the scope of this document.`

目前仅仅谈论trace encoder的控制需求，尚不谈论这些域的组织方式和访问方式

之后可以参考https://github.com/riscv-non-isa/tg-nexus-trace/blob/master/docs/RISC-V-Trace-Control-Interface.adoc#register-map实现组织和访问方式

为尽快熟悉流程，先仅考虑M（Mandatory），指令trace的必做部分

problem 6

p19

`Other abbreviations used in the tables are:`
`W column heading indicates field width (in bits)`

`G column heading indicates field group`

`RW column heading indicates whether bit is read-only (R), or read-write (RW). For the latter, it is allowed but not required that the bit be writable`
`Rst column heading indicates field reset value. SD in this column indicates a system dependent reset value`

- W:域的宽度
- G:所属的组，目前仅考虑M
- RW:读/读写
- Rst:复位的值，SD表示系统依赖（system dependent）复位值

problem 7

p21

field	W	G	RW	Rst
Active	1	M	RW	0
teEnable	1	M	RW	0
iTracing	1	M	RW	0
ResyncMode	2	M	RW	SD

- Active:

  `Primary enable for trace system. When 0, the trace system may have clocks gated off or be powered down, and other register locations may be inaccessible. Hardware may take arbitrarily long to process powerup or power-down and will indicate completion when the read value of this bit matches the value written.`

  trace系统的主使能。

  0:什么叫做clocks gated off?其他寄存器位置不可访问?

  什么叫做读值和写值匹配时？WARL, WLRL?

- teEnable:

  `Primary trace enable. Trace can be enabled via iTracing or dTracing when 1. Setting to 0 flushes any queued trace data to the designated sink.`

  主trace使能。

  1:可以启用指令trace或者数据trace

  0:是指输出成packet？还是丢弃?

- iTracing:

  `Instruction trace enable. When 1, trace will be generated, subject to any optional filtering. May be written by software, or via triggers if iTrigEnable is 1.`

  指令trace使能。

  1:启动指令trace

  什么是受任何可选筛选影响?

  软件可写？或者通过trigger如果iTriEnable为1

- ResyncMode:

  选择重同步机制，至少需要实现一种非零的

0:关闭

1:计数packet

2:计数时钟周期

3:计数instruction (16-bit) half-words，指令的半字数，C扩展计数1，其他指令计数2。

problem 8

p28

- 顺序指令：不需要trace
- 不可推断的PC不连续：需要trace下一条有效地址。jalr, Interrupts and exceptions
- 分支：需要trace是否跳转；jal不需要
- 中断/异常：中断是异步的，而异常可以和具体的某条指令相关。需要trace中断前的最后一条指令，并给出中断的目标（可以仅仅给出异常类型）。需要同时追踪两条指令，异常前最后一条指令，下一条有效指令。并且并不是所有的中断/异常都会引起PC变化，仅仅trace trap
- 同步：同时是通过发送一个完整值的指令地址（可能还有一个上下文标识）来完成的?

  复位后第一条指令

  任何时间一条指令被trace而前一条指令未被trace

  中断/异常 handler

  经过一段很长的时间
- trace终止

  输出最后一条trace的指令

problem 9

p30

先仅仅支持Delta address mode

其不trace完整的指令地址PC，而是trace当前指令和之前包含地址的一个数据包中的指令地址的差值。

problem 10

p35

Mandatory的信息 从核心到encoder （interface）

- 总共退休的？正在退休的？指令数目
- 异常和中断的原因和trap值

  ucause/scause/vscause/mcause

  utval/stval/vstval/mtval

  只输出某个特权集的?
- 核心当前的特权等级
- 退休指令的类型

jalr

b

mret sret

- 指令地址

  jalr

  jalr->next

  b

  中断/异常 前后两条指令

  特权等级切换 前后两条指令

problem 11

p38

一个core可能会有多个hart

- 为每个hart实现一个接口
- 为core实现一个接口，然后内部选择连接哪个hart

  后者由于线程切换而发生的频繁的上下文切换会导致编码效率极低，不太建议

problem 12

p38

此处仅考虑

- M Mandatory
- MR（Mandatory， may be replicated: `For harts that can retire a maximum of N "special" instructions per clock cycle, the interface must include N instances of this signal.` 每个时钟周期可以回收多条吗？特殊指令是指什么，需要trace的吗?

  `"Special" instructions are those that require itype to be non-zero`
- BR （Block, may be replicated): `Mandatory for harts that can retire multiple instructions in a block. Replication as per OR. If omitted, the interface must include SR group signals instead.`
- SR （Single, may be replicated): `Mandatory for harts that can only retire one instruction in a block. Replication as per OR (see section 4.2.2). If omitted, the interface must include BR group signals instead` 什么叫做per block?

Signal	Group
itype[itype_width_p-1:0]	MR
cause[ecause_width_p-1:0]	M
tval[iaddress_width_p-1:0]	M

Signal	Group
priv[privilege_width_p-1:0]	M
iaddr[iaddress_width_p-1:0]	MR
iretire[iretire_width_p-1:0]	BR
ilastsize[ilastsize_width_p-1:0]	BR
iretire[0:0]	SR
ilastsize[ilastsize_width_p-1:0]	SR

- itype

  `Termination type of the instruction block`

  指令块的终止类型?

Value	描述
0	当前块的最后一条指令，且?不是其它的类型
1	异常，在当前块最后一条退休后发生
2	中断，在当前块最后一条退休后发生
3	异常/中断返回
4	分支不跳转
5	分支跳转
6	itype_width_p==3？ 不可推断的跳转： 保留
7	保留
8	不可推断的call
9	可推断的call
10	不可推断的尾调用tail-call
11	可推断的尾调用
12	`Co-routine swap`?
13	返回
14	其他不可推断的跳转
15	其他可推断的跳转

- cause

中断/异常的原因

ucause/scause/vscause/mcause

只有itype为1/2才有用

- tval

  trap value

  utval/stval/vstval/mtval

  只有itype为1有用

- priv

  这个时钟退休的所有指令的特权等级?

Value	描述
0	U
1	S/HS
2	保留
3	M
4	D
5	VU
6	VS
7	保留

- iaddr

  这个块中第一条退休的指令

  Invalid 如果 iretire=0, 除非itype=1.这种情况下，指的是引起异常的指令

- iretire(BR)

  由该块中退出的指令表示的半字数

- ilastsize(BR)

  最后一个退出指令的大小是$2^{ilastsize}$的半字

- iretire(SR)

  该块中退休的指令（0/1）

- ilastsize(SR)

  退出指令的大小是$2^{ilastsize}$的半字

补充说明

- `The information presented in a block represents a contiguous block of instructions starting at iaddr, all of which retired in the same cycle`

  在一个块中显示的信息表示一个从iaddr开始的连续指令块，所有这些指令都在同一个周期中退出

- 如果itype==1/2(异常/中断)，则退休指令数可能为0
- itype==1/2时cause/tval才有定义
- 如果iretire==0 && itype==0，则所有信号都无意义
- 多退休的情况

  `For harts that can retire a maximum of N non-zero itype values per clock cycle, the signal groups MR, OR and either BR or SR must be replicated N times. Typically N is determined by the maximum number of branches that can be retired per clock cycle. Signal group 0 represents information about the oldest instruction block, and group N-1 represents the newest instruction block. The interface supports no more than one privilege change, context change, exception or interrupt per cycle and so signals in groups M and O are not replicated. Furthermore, itype can only take the value 1 or 2 in one of the signal groups, and this must be the newest valid group (i.e. iretire and itype must be zero for higher numbered groups). If fewer than N groups are required in a cycle, then lower numbered groups must be used first. For example, if there is one branch, use only group 0, if there are two branches, instructions up to the 1st branch must be reported in group 0 and instructions up to the 2nd branch must be reported in group 1 and so on`

  对于每个时钟周期最多可以收回N个非零itype值的hart，信号组MR, OR和BR或SR必须被重复N次。

  通常N由每个时钟周期中可以被撤销的分支的最大数目决定。

  信号组0表示最早的指令块信息，信号组N-1表示最新的指令块信息。

  接口每个周期只支持一次特权更改、上下文更改、异常或中断，因此M组和O组中的信号不会被重复。

  此外，itype只能在其中一个信号组中取1或2的值，并且这必须是最新的有效组(即iretire和itype必须为零对于更高数目的组？)

  先使用编号较低的组
- 单退休的情况

  iretire 0/1

  ilastsize仍然需要，隐返回模式需要用它计算预测地址？