Analysis of RISC-V Vector Performance using MCA Tools

Michael Maitland
**SIMD and MCA**

- Most instructions can operate on different data types, specified in the instruction encoding.
- Software indicates the size required by appending a suffix to the instruction mnemonic.
- The number of elements operated on is indicated by the specified register size.

**Neon:** VADD.I16 q0, q1, q2

- 16-bit integer elements
- Elements stored in 128-bit Q registers
- Eight 16-bit lanes in parallel

**x86:** VADDPD ZMM0, ZMM1, ZMM2

- 64-bit doubles elements
- Elements stored in 512-bit ZMM registers
- Eight 64-bit lanes in parallel

**RISC-V Vectors and MCA**

- Most instructions can operate on different data types, specified in the instruction encoding.
- Software indicates the size required by setting the selected element width (SEW) by calling vset{i}vl{i} to modify vtype.
- The number of elements operated on is indicated by setting the vector register grouping (LMUL) and vector length (VL) by calling vset{i}vl{i} to modify vtype.

[1]: https://developer.arm.com/documentation/dht0002/a/Introducing-NEON/NEON-architecture-overview/NEON-instructions

[2]: https://github.com/riscv/riscv-v-spec/blob/master/v-spec.adoc
The information required to know the scheduling for a SIMD instruction is encoded in the instruction itself, but for a RISC-V instruction it is split between the instruction and global registers.
The Problem

1. vadd.vv v12, v12, v12

2. vsetvli zero, a0, e8, m1, tu, mu
   vadd.vv v12, v12, v12
   vsetvli zero, a0, e8, m8, tu, mu
   vadd.vv v12, v12, v12

3. vsetvli zero, a0, e8, m1, tu, mu
   vadd.vv v12, v12, v12

4. vsetvl rd, rs1, rs2
   vadd.vv v12, v12, v12
   vsetvl rd, rs1, rs2
   vadd.vv v12, v12, v12

No vset{vi}vl{vi} to construct vi and vtype state.

MCA does not track state like vl or vtype.

vl and vtype change state during program.

Sometimes it's not possible to use just the vset{vi}vl{vi} instruction to construct vi or vtype because no immediate.
The Solution

Instrument Regions

Note: Patch up for review and looking for feedback!
Impact

● Meaningful MCA reports for programs or regions containing RVV instructions
● Create new kinds of instruments for instructions that depend on other kinds of global state
  ○ SEW and VLEN
  ○ CSR registers

What’s Next?

● LLVM knows LMUL, VL, SEW at compile time
  ○ Generate asm with instrument comments
● LLVM-MCAD knows what is in the vtypei register at runtime
  ○ LLVM-MCAD can inject instrument comments before sending to MCA
● Evaluation of LLVM-MCAD with instruments compared to true cycle count