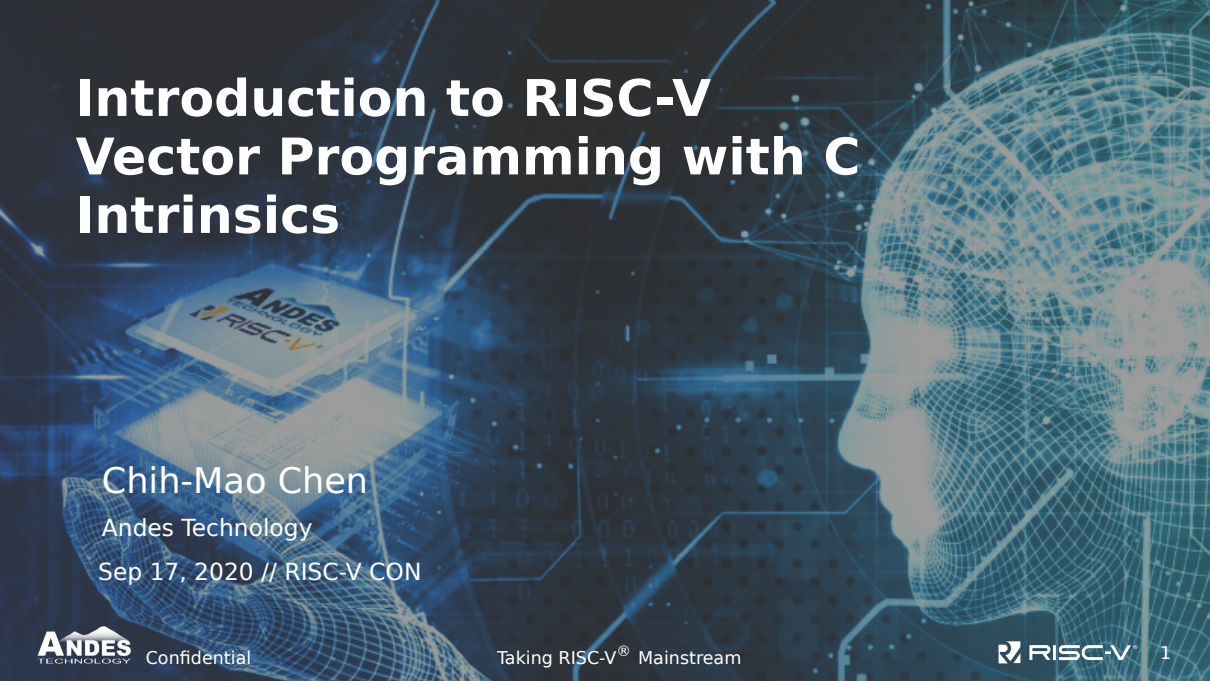
Introduction to RISC-V Vector Programming with C Intrinsics



Chih-Mao Chen
Andes Technology
Sep 17, 2020 // RISC-V CON

RISC-V Vector Extension

- Implementation-defined vector register length (VLEN)
- Variable element width and vector length
- Vector register grouping
- Polymorphic & maskable vector instructions
- Vector-length agnostic programming model

RVV Intrinsics

- Implement complex vectorized algorithm
- Better integration with C code
- Porting existing vector libraries (e.g. SLEEF)

RVV Intrinsics

- Implement complex vectorized algorithm
- Better integration with C code
- Porting existing vector libraries (e.g. SLEEF)

- How to work with RVV scalable vectors?
- RFC: https://github.com/riscv/rvv-intrinsic-doc

RVV Intrinsic Types

Vector types

v**type**m**lmul**_t

- `vint8m1_t`
- `vfloat32m8_t`
- `vint16mf2_t`

Vector mask types

vbool**mlen**_t

- `MLEN = SEW/LMUL`
- `vbool1_t`
- `vbool2_t`
- `...`
- `vbool64_t`

RVV Intrinsic Functions

- Each RVV intrinsic function maps to one RVV instruction for a specific SEW/LMUL

```
vint8m1_t vadd_vv_i8m1(vint8m1_t vs2, vint8m1_t vs1);
vint32m1_t vadd_vx_i32m1(vint32m1_t vs2, int32_t rs1);
vfloat64m2_t vfwmul_vv_f64m2 (vfloat32m1_t vs2, vfloat32m1_t vs1);
```

- Intrinsic functions sets up the vtype via `vsetvli x0, x0, <vtype>`
  - Eliminated by compiler if vtype did not change
- Implicit active vector length (VL)
- Intrinsic function overloading

Masked RVV Intrinsic Functions

- Most RVV intrinsic functions also take an optional `mask` and `maskedoff` argument:

```
// vd[i] = mask[i] ? vop(vs1[i], vs2[i]) : maskedoff[i]
vint8m1_t vadd_vv_i8m1_m(
  vbool8_t mask, vint8m1_t maskedoff, vint8m1_t vs2, vint8m1_t vs1
);
```

Strip-mining

```
#include <riscv_vector.h>

void vector_add(size_t n, uint32_t a[n],
                const uint32_t b[n], const uint32_t c[n]) {
  while (n > 0) {
    size_t vl = vsetvl_e32m8(n);
    vuint32m8_t vb = vle32_v_u32m8(b);
    vuint32m8_t vc = vle32_v_u32m8(c);
    vuint32m8_t va = vadd(vb, vc);
    vse32_v_u32m8(a, va);
    a += vl; b += vl; c += vl; n -= vl;
  }
}
```

Strip-mining

```
vector_add:                             # @vector_add %entry
        beqz    a3, .LBB0_2
.LBB0_1:                                # %while.body
        vsetvli a4, a3, e32,m8
        vle32.v v8, (a1)
        vle32.v v16, (a2)
        vadd.vv v8, v8, v16
        vse32.v v8, (a0)
        slli    a5, a4, 2
        add     a0, a0, a5
        add     a1, a1, a5
        sub     a3, a3, a4
        add     a2, a2, a5
        bnez    a3, .LBB0_1
.LBB0_2:                                # %while.end
        ret
```

Andes RVV Compiler

- LLVM/Clang-based C/C++ compiler for Andes NX27V (RV64GCNPV)
- Full support of RVV intrinsic
- Inline assembly support
- Tracks the latest RVV standard
- `vsetvl` elimination
- Scalable vector register allocation (and spilling)

Thank you