SiFive

Accelerating the migration from ARM NEON to RISC-V Vectors

Han-Kuan Chen
Senior Engineer, SiFive

Outline

- What is intrinsics?
- How do software support various intrinsics?
- SiFive Recode
- Improve SiFive Recode
- Arm Compute Library benchmark
- OpenCV benchmark

Acknowledgments

Special thanks to Craig Topper, Kito Cheng, Peter Liao and Yi-Hsiu Hsu, who provided mentorship and guidance.

What is intrinsics?

- Intrinsics are low-level functions provided by compiler that allow direct access to specific CPU instructions.
- Directly using intrinsics leverages hardware capabilities, which improves execution speed of performance-critical software tasks.
- Most major vendors (Intel, AMD, ARM, etc.) offer intrinsics.
  - x86: SSE & AVX
  - arm: NEON
  - RISC-V: RVV
- Intrinsics are widely used in software.
  - e.g., TensorFlow, Arm Compute Library, OpenCV, libyuv …

How do software support various intrinsics?

SiFive

- Due to the presence of various intrinsics, some projects have been proposed to minimize the effort required for porting.
- Provide an universal interface and translate it to different targets.
  - e.g., xnnpack and highway
- Transfer intrinsics internally into another different intrinsics.
  - e.g., simde, AvxToNeon, neon2sse and sse2neon
- RISC-V is new, how do we support various software and intrinsics?

SiFive Recode

- Protect your existing software investment, migrate with confidence.

```
#include <arm_neon.h>

float32_t dot_prod(
  const float32_t *in1, const float32_t *in2, uint32_t blockSize) {
 float32x4_t acc = vdupq_n_f32(0.0f);

 for (uint32_t i = 0; i != blockSize; i += 4) {
   float32x4_t A = vld1q_f32(in1 + i);
   float32x4_t B = vld1q_f32(in2 + i);
   acc = vmlaq_f32(acc, A, B);
 }

 return vaddvq_f32(acc);
}
```

ARM NEON Source code

Compiler option

`-msifive-recode=neon`

```
dot_prod:
        beqz    a2, .LBB0_3
        vsetivli        zero, 4, e32, mf2, ta, ma
        vmv.v.i v8, 0
.LBB0_2:
        vle32.v v9, (a0)
        addi    a0, a0, 16
        vle32.v v10, (a1)
        addi    a1, a1, 16
        vfmacc.vv       v8, v9, v10
        addiw   a2, a2, -4
        bnez    a2, .LBB0_2
        j       .LBB0_4
.LBB0_3:
        vsetivli        zero, 4, e32, mf2, ta, ma
        vmv.v.i v8, 0
.LBB0_4:
        li      a0, 32
        vsetivli        zero, 2, e32, mf2, ta, ma
        vnsrl.wi        v9, v8, 0
        vnsrl.wx        v8, v8, a0
        vfadd.vv        v8, v9, v8
        vsetivli        zero, 1, e32, mf2, ta, ma
        vslidedown.vi   v9, v8, 1
        vfadd.vv        v8, v8, v9
        vfmv.f.s        fa0, v8
        ret
```

Compiled SiFive Assembly Code

Improve SiFive Recode

- We optimize a header based Recode to LLVM based Recode.
  - RVV intrinsics in header versus lowering NEON intrinsics into LLVM IR
- The issues of header based Recode
  - Hard to improve the performance.
  - Hard to maintain.
- The advantages of LLVM based Recode
  - LLVM has rich optimizations.

Lower NEON intrinsics into LLVM IR

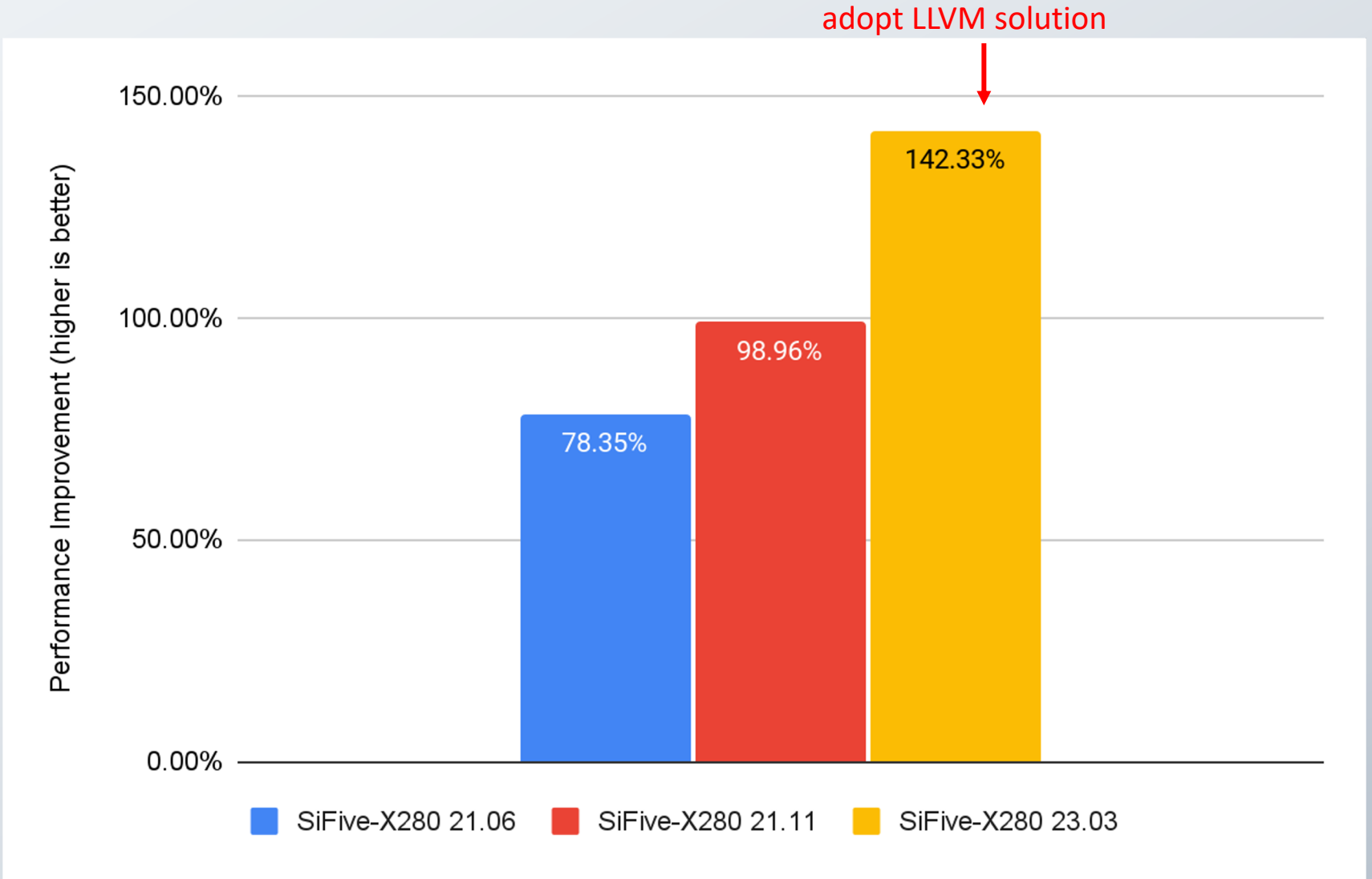
NEON intrinsics	LLVM IR

define void @vabsq_s8(ptr %in_0, ptr %out) {
entry:
  %0 = load <16 x i8>, ptr %in_0, align 1
  %2 = tail call <16 x i8> @llvm.aarch64.neon.abs.v16i8(<16 x i8> %0)
  store <16 x i8> %2, ptr %out, align 1
  ret void
}
	define void @vabsq_s8(ptr %in_0, ptr %out) {
	entry:
	  %0 = load <16 x i8>, ptr %in_0, align 1
	  %1 = call <16 x i8> @llvm.abs.v16i8(<16 x i8> %0, i1 false)
	  store <16 x i8> %1, ptr %out, align 1
	  ret void
	}

define void @vmull_u8(ptr %in_0, ptr %in_1, ptr %out) {
entry:
  %0 = load <8 x i8>, ptr %in_0, align 1
  %1 = load <8 x i8>, ptr %in_1, align 1
  %2 = tail call <8 x i16> @llvm.aarch64.neon.umull.v8i16(<8 x i8> %0, <8 x i8> %1)
  store <8 x i16> %2, ptr %out, align 2
  ret void
}
	define void @vmull_u8(ptr %in_0, ptr %in_1, ptr %out) {
	entry:
	  %0 = load <8 x i8>, ptr %in_0, align 1
	  %1 = load <8 x i8>, ptr %in_1, align 1
	  %2 = zext <8 x i8> %0 to <8 x i16>
	  %3 = zext <8 x i8> %1 to <8 x i16>
	  %4 = mul nuw <8 x i16> %2, %3
	  store <8 x i16> %4, ptr %out, align 2
	  ret void
	}

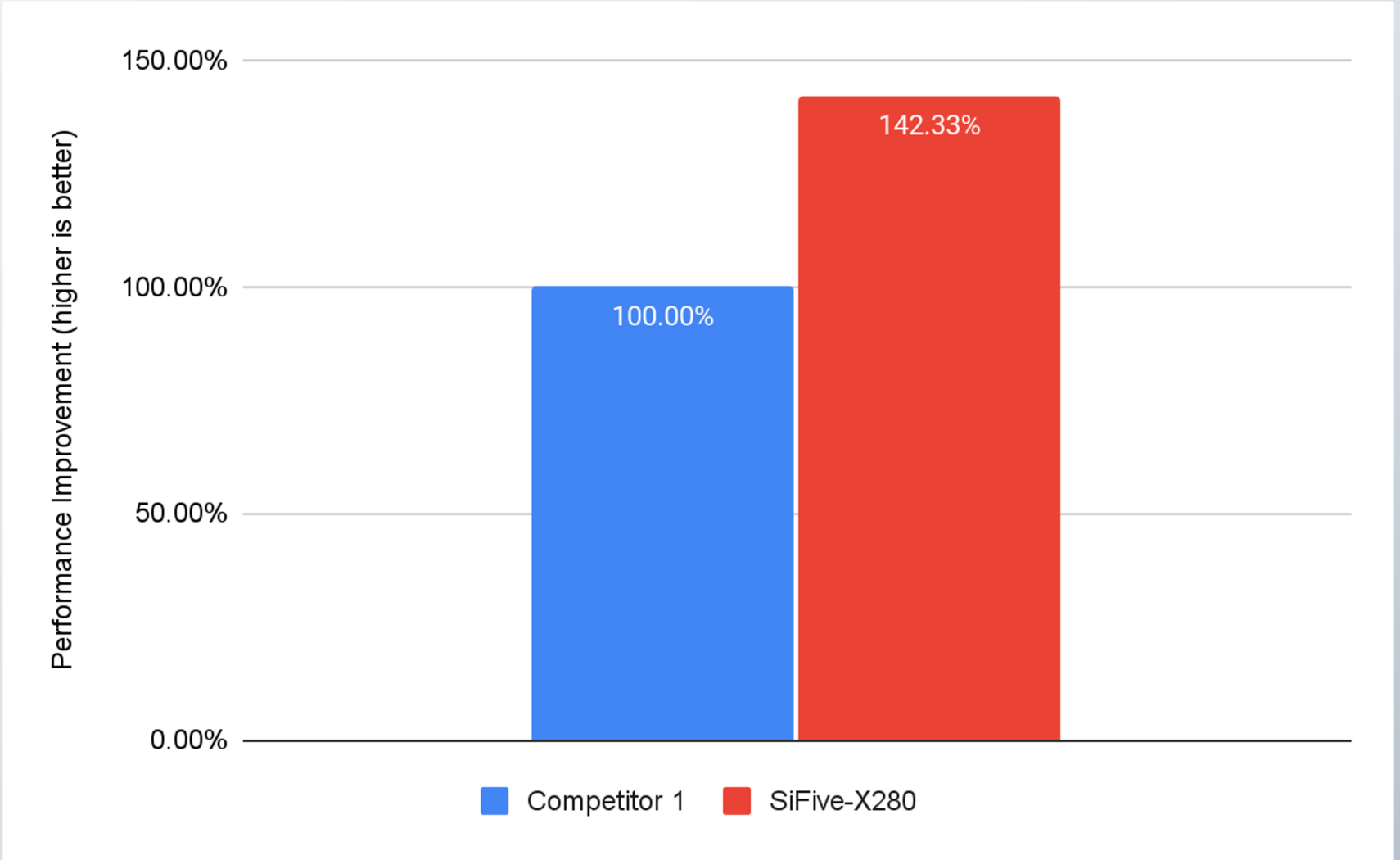
header based Recode VS LLVM based Recode

- 43 % speedup after using LLVM solution



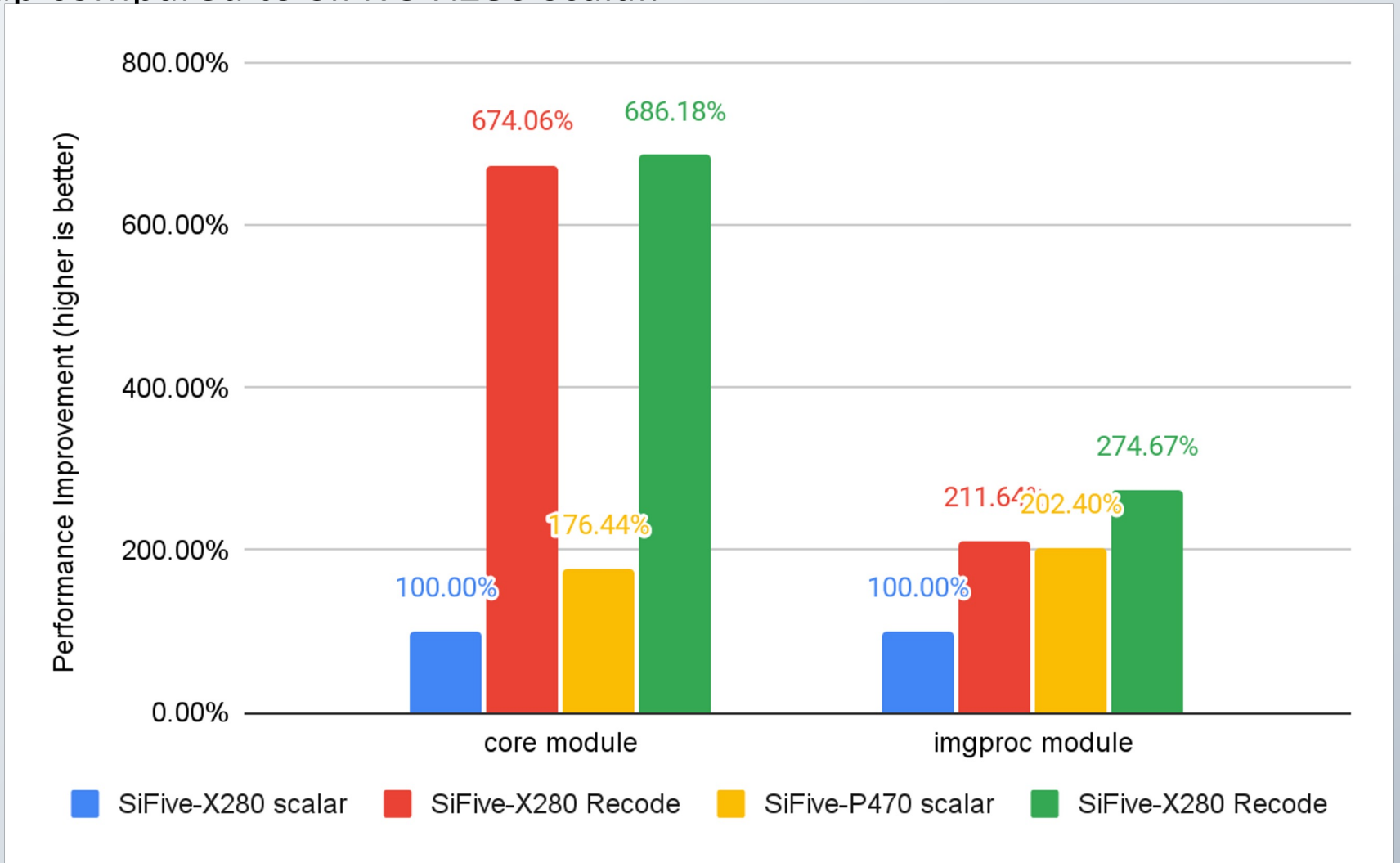
Arm Compute Library benchmark

- 42 % average speedup compared to the competitor.



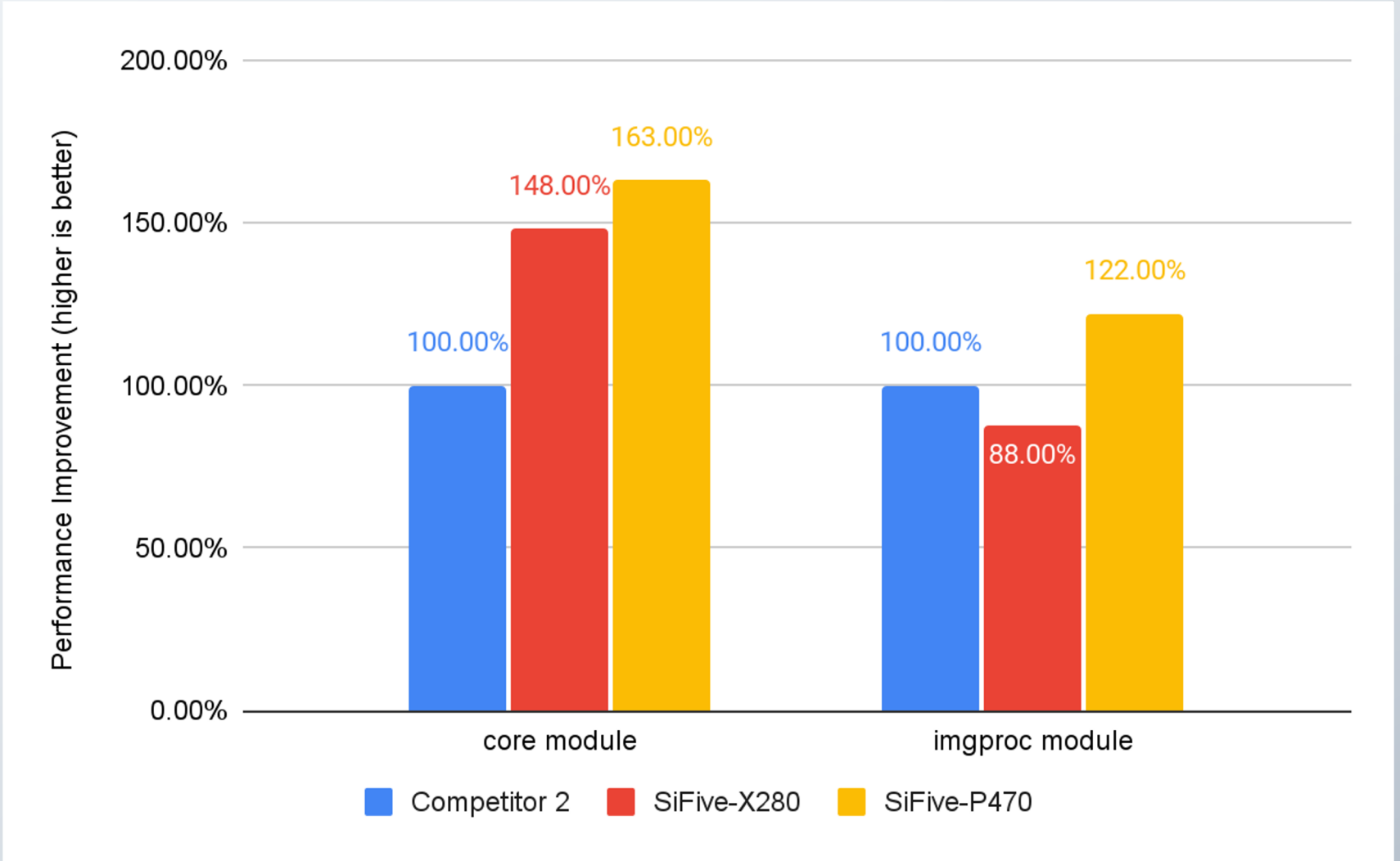
OpenCV benchmark

- In core module, SiFive-X280 Recode is 6.7x speedup and SiFive-P470 Recode is 6.9x speedup compared to SiFive-X280 scalar.



OpenCV benchmark (Cont.)

- In core module, SiFive-P470 Recode is 1.6x speedup compared to the competitor.



Is it the end?

- No. We strive for higher performance.
  - We are investing new optimization. At least 20% improvement is expected.
- No. Recode will support more intrinsics.
  - SSE, AVX and SVE will be supported.

Empowering innovators

www.sifive.com