LLVM Support for Sub-FP8 Quantization with RISC-V Extensions for Machine Learning Models

Jhih-Kuan Lin, Fu-Jian Shen, Kathryn Chapman, Mengshiun Yu, Jenq-Kuen Lee
Department of Computer Science, National Tsing-Hua University, Taiwan

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Motivation

- **Model quantization** optimizes performance by reducing 32-/16-bit INT/FP formats to 8-, 6-, and 4-bit formats, lowering compute, memory, and transmission costs.
- The Open Compute Project (OCP) defines floating-point formats smaller than 8-bit. We developed the **sub-FP8 ISA** extension for RISC-V to enable support for these formats in scalar operations.

Sub-FP8 Support in LLVM

- Sub-FP8 Backend & Code Generation
- Instruction Support for Sub-FP8 in Assembler
- Efficient Pattern Matching for Sub-FP8 Operations
- Sub-FP8 C/C++ Intrinsics

```
__builtin_riscv_sfadd_f8e4m3_f8e4m3(0.1f8e4m3, 0.1f8e4m3);
```

⇩

```
%0 = call f8e4m3 @llvm.riscv.pllab.sfadd.f8e4m3.f8e4m3.f8e4m3(f8e4m3 0xQ1D, f8e4m3 0xQ1D)
```

⇩

```
f8e4m3 = llvm.riscv.pllab.sfadd TargetConstant:i64<9897>, ConstantFP:f8e4m3<APFloat(29)>, ConstantFP:f8e4m3<APFloat(29)>
```

⇩

```
PseudoSSET_INSERT 1, 1, implicit-def $eft, implicit-def $sft
renamable $f15_s = nofpexcept PseudoSFADD killed renamable $f15_s, renamable $f15_s, 7, 1, 1, implicit killed $eft, implicit killed $sft, implicit $frm
```

⇩

```
csrwi   sft, 1
csrwi   eft, 1
sfadd   fa5, fa5, fa5
```

Fig. 2. Sub-FP8 Compile Flow in LLVM (from C to assembly)

Simulation

- The instruction set architecture (ISA) was simulated, allowing for correctness verification, using **Spike.**
- **Gem5** was used to execute real programs and collect research data.

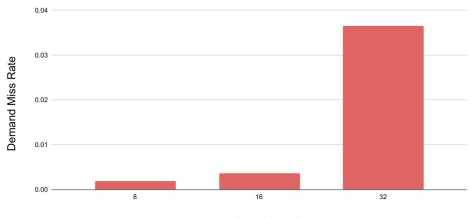

Fig. 3. The results of GEMM computation ((256×16) × (16×256)) under an environment with a 32KB L1 D-cache and compilation optimization level O3. Cache miss rates across different floating-point formats are also compared.

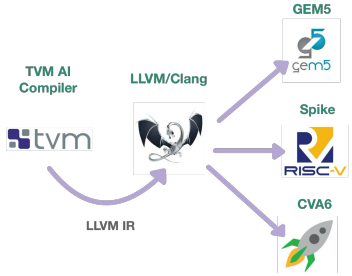

Fig. 4. Overall architecture of our Sub-FP8 compilation and simulation flow.

EFT & SFT

ssft/seft[3:0]				SFT/EFT
0	0	0	1	FP8(E5M2)
0	0	1	0	FP8(E4M3)
0	0	1	1	FP6(E3M2)
0	1	0	0	FP6(E2M3)
0	1	0	1	FP4(E2M1)
others				reserve

- The FCSR was modified to hold additional fields, SFT and EFT, for type information.
- Performing data flow analysis for **automatic SSET insertion** to reduce redundant sset instructions.

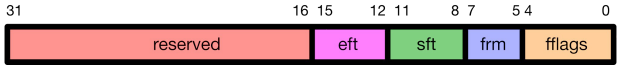

Fig. 1. Contents of the modified Floating-Point Control and Status Register (FCSR)

RISC-V Vector Extensions

- The following vector extension instructions are proposed to support the sub-FP8 format:

Class	Instruction name	Description
Arithmetic	vsfadd, vsfsub, vsfmul, vsfdiv, vsfmacc, vsfsqrt	Vector-scalar addition, subtraction, multiplication, division,and multiply-accumulate
Comparison	vsfmin, vsfmax	Compute element-wise min/max
Reduction	vsfredosum, vsfredusum, vsfredmin, vsfredmax, vsdot	Perform reduction operations on a vector, such as sum or min/max, to produce a scalar result
Data Movement	vfmv,vfcvt(float -> float)	Move or convert data
Other	vsfsgnjn, vsfclass,	Other operations for floating-point numbers

Example for GEMM

- C code snippet demonstrating a **GEMM operation** using the newly added RISC-V f8e4m3 floating-point format.

```
for (int i = 0; i < MLEN; ++i)
    for (int j = 0; j < NLEN; ++j)
      for (int k = 0; k < KLEN; ++k)
        {
          golden_array[i * NLEN + j] =
__builtin_riscv_sfmadd_f8e4m3_f8e4m3(a_array[i * KLEN + k],
b_array[j + k * NLEN], golden_array[i * NLEN + j]);
        }
```

Hardware Reference Design Based on CVA6

- RISC-V **CVA6** Core (CV64A6) (Work-in-Progress) using the transprecision-supporting FPU, CVFPU (FPnew)

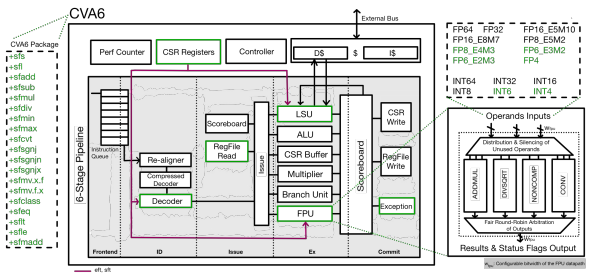

Fig. 5. CVA6 processor modified to support the sub-FP8 format and accompanying instructions. The red line shows the stages of the pipeline that use the format information provided by the FCSR (*sset*). New features and modules with significant modifications are shown in green.

Contact: kjchapman@pllab.cs.nthu.edu