RISC-V Audiomark Coding Task

By: BhuvanB

Problem Statement

This document describes the design, implementation, and performance analysis of a RISC-V Vector Extension accelerated version of a Q15 saturating multiply-accumulate function.

Implement the following function in C:

```
void q15_axpy_rvv(const int16_t *a, const int16_t *b, int16_t
*y, int n, int16_t alpha);
```

That computes, for all i in [0..n):

$$y[i] = sat\_q15(a[i] + \alpha \times b[i])$$

Implementation Overview

Scalar Reference

The scalar implementation performs:

1. Q15 × Q15 multiply
2. Accumulation with a Q15 addend
3. Saturation to the int16_t range

This version is used as a reference for verification.

RVV Vectorized Implementation

The RVV implementation uses:

1. Vector-length agnostic loops via vsetvl
2. 16-bit vector load (vle16)
3. Widening multiply (vwmul) for α × b
4. Widening accumulation in 32-bit lanes
5. Saturation and narrowing back to Q15 using vnclip
6. Vector store (vse16)

Design Choices

1. Widening arithmetic is used to prevent overflow during multiplication and accumulation
2. Explicit saturation ensures bit-for-bit equivalence with the scalar reference
3. The implementation is vector-length agnostic using vsetvl

Build and Run Environment

Building

```
make
```

Running

Spike RISC-V ISA Simulator:

```
spike --isa=rv32gcv_zicntr pk ./vector.elf
```

Results

```
$ spike --isa=rv32gcv_zicntr pk ./vector.elf

Cycles ref: 56833

Verify RVV: OK (max diff = 0)

Cycles RVV: 5072
```

Speedup : 11.2x

Implementation Repository

GitHub