



Matrix Multiply Accumulate Instruction (Xsfvfmacc- qqq) Extension Specification

Version 1.0

Copyright © by SiFive, Inc. All rights reserved.

Matrix Multiply Accumulate Instruction (Xsfvfwmacccqq) Extension Specification

Proprietary Notice

Copyright © 2023 by SiFive, Inc. All rights reserved.

Information in this document is provided “as is,” with all faults.

SiFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

SiFive does not assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

SiFive reserves the right to make changes without further notice to any products herein.

Revision History

Version	Date	Changes
1.0	September 7, 2023	Initial release

Contents

1 Xsfvfwmacccqq Extension: Matrix Multiply Accumulate Instruction	2
1.1 C Intrinsic	3

1

Xsfvfwmacccqq Extension: Matrix Multiply Accumulate Instruction

The SiFive custom vector extension **Xsfvfwmacccqq** introduces the Bfloat16 matrix multiply accumulate instruction. The assembly mnemonic for this instruction is:

```
sf.vfwmacc.4x4x4 vd, vs1, vs2    # funct6=111100
```

Note

Similar to 1D vector fused-multiply-add instructions, and unlike most vector assembly mnemonics, the *vs1* operand is given first to match the layout of the corresponding mathematical expression.

The *sf.v* prefix indicates that this is a SiFive custom vector instruction. The *f* indicates a floating-point operation. The *w* stands for *widening*. The *macc* is for *multiply-accumulate*. The *MxKxN* suffix indicates that *vs1* represents an *MxK* matrix, *vs2* represents *KxN* matrices, and *vd* represents *MxN* matrices—in this case, all 4x4.

The instruction is encoded similarly to other vector instructions, but within the custom-2 major opcode, with *funct3*=1 (OPFVV), and with *funct6* as specified above.

The instruction performs *vL*/16 matrix-multiply and accumulate operations:

```
vd[0][4][4]      += vs1[0][4][4] * vs2[0][4][4]
vd[1][4][4]      += vs1[0][4][4] * vs2[1][4][4]
...
vd[(vL/16)-1][4][4] += vs1[0][4][4] * vs2[(vL/16)-1][4][4]
```

Note that only the first tile in *vs1* is used to multiply each tile of *vs2* and accumulate into corresponding tiles of *vd*.

vs1 has EMUL=1, vs2 has EMUL=LMUL, and vd has EMUL=2*LMUL.

The instruction is reserved if v1 is not divisible by 16, or if SEW≠16, or if masked.

vd must not overlap vs1.

Additional tiles are stored contiguously in the vector register group, and multiple tiles can be operated on in parallel in a single matrix multiply instruction.

1.1 C Intrinsics

A full list of C intrinsics compatible with Xsfvfmaccqqq are provided below:

```
vfloat32mf2_t __riscv_sf_vfmacc_4x4x4_f32mf2(vfloat32mf2_t vd, vbfloating16m1_t vs1,
vbfloating16mf4_t vs2, size_t vl)
vfloat32m1_t __riscv_sf_vfmacc_4x4x4_f32m1(vfloat32m1_t vd, vbfloating16m1_t vs1, vbfloating16mf2_t
vs2, size_t vl)
vfloat32m2_t __riscv_sf_vfmacc_4x4x4_f32m2(vfloat32m2_t vd, vbfloating16m1_t vs1, vbfloating16m1_t
vs2, size_t vl)
vfloat32m4_t __riscv_sf_vfmacc_4x4x4_f32m4(vfloat32m4_t vd, vbfloating16m1_t vs1, vbfloating16m2_t
vs2, size_t vl)
vfloat32m8_t __riscv_sf_vfmacc_4x4x4_f32m8(vfloat32m8_t vd, vbfloating16m1_t vs1, vbfloating16m4_t
vs2, size_t vl)

vfloat32mf2_t __riscv_sf_vfmacc_4x4x4(vfloat32mf2_t vd, vbfloating16m1_t vs1, vbfloating16mf4_t vs2,
size_t vl)
vfloat32m1_t __riscv_sf_vfmacc_4x4x4(vfloat32m1_t vd, vbfloating16m1_t vs1, vbfloating16mf2_t vs2,
size_t vl)
vfloat32m2_t __riscv_sf_vfmacc_4x4x4(vfloat32m2_t vd, vbfloating16m1_t vs1, vbfloating16m1_t vs2,
size_t vl)
vfloat32m4_t __riscv_sf_vfmacc_4x4x4(vfloat32m4_t vd, vbfloating16m1_t vs1, vbfloating16m2_t vs2,
size_t vl)
vfloat32m8_t __riscv_sf_vfmacc_4x4x4(vfloat32m8_t vd, vbfloating16m1_t vs1, vbfloating16m4_t vs2,
size_t vl)

vfloat32mf2_t __riscv_sf_vfmacc_4x4x4_f32mf2_tu(vfloat32mf2_t vd, vbfloating16m1_t vs1,
vbfloating16mf4_t vs2, size_t vl)
vfloat32m1_t __riscv_sf_vfmacc_4x4x4_f32m1_tu(vfloat32m1_t vd, vbfloating16m1_t vs1,
vbfloating16mf2_t vs2, size_t vl)
vfloat32m2_t __riscv_sf_vfmacc_4x4x4_f32m2_tu(vfloat32m2_t vd, vbfloating16m1_t vs1,
vbfloating16m1_t vs2, size_t vl)
vfloat32m4_t __riscv_sf_vfmacc_4x4x4_f32m4_tu(vfloat32m4_t vd, vbfloating16m1_t vs1,
vbfloating16m2_t vs2, size_t vl)
vfloat32m8_t __riscv_sf_vfmacc_4x4x4_f32m8_tu(vfloat32m8_t vd, vbfloating16m1_t vs1,
vbfloating16m4_t vs2, size_t vl)

vfloat32mf2_t __riscv_sf_vfmacc_4x4x4_f32mf2_tu(vfloat32mf2_t vd, vbfloating16m1_t vs1,
vbfloating16mf4_t vs2, size_t vl)
vfloat32m1_t __riscv_sf_vfmacc_4x4x4_f32m1_tu(vfloat32m1_t vd, vbfloating16m1_t vs1,
vbfloating16mf2_t vs2, size_t vl)
vfloat32m2_t __riscv_sf_vfmacc_4x4x4_f32m2_tu(vfloat32m2_t vd, vbfloating16m1_t vs1,
vbfloating16m1_t vs2, size_t vl)
vfloat32m4_t __riscv_sf_vfmacc_4x4x4_f32m4_tu(vfloat32m4_t vd, vbfloating16m1_t vs1,
```

Matrix Multiply Accumulate Instruction (Xsfvfmaccqqq) Extension Specification
Xsfvfmaccqqq Extension: Matrix Multiply Accumulate Instruction

```
vbfloat16m2_t vs2, size_t vl)  
vfloat32m8_t __riscv_sf_vfmacc_4x4x4_f32m8_tu(vfloat32m8_t vd, vbfloat16m1_t vs1,  
vbfloat16m4_t vs2, size_t vl)
```