SiFive Int8 Matrix Multiplication Extensions Specification

Version 1.1

SiFive Int8 Matrix Multiplication Extensions Specification

Proprietary Notice

Revision History

Version	Date	Changes
1.1	November 28, 2023	• Fixed the intrinsic list
1.0	September 7, 2023	• Initial release

Contents

1

SiFive Custom Instructions for Matrix Multiplication

This document defines the Xsfvqmaccqoq and Xsfvqmaccdod extensions, which provides support for Int8 Matrix Multiplication Instructions.

1.1 Xsfvqmaccqoq Extension: Int8 Matrix Multiplication Instructions (4-by-8 and 8-by-4)

These instructions are signed or unsigned integer matrix-multiply-add instructions that widen 8-bit integer input values to 32-bits. To understand the functionality of these instructions, consider the arithmetic below:

```
# Dest (C)   Input (A)  *  Input (B)
  int32 +=       uint8  *      uint8
  int32 +=       uint8  *       int8
  int32 +=        int8  *      uint8
  int32 +=        int8  *       int8
```

Here, the (u)int8 A- and B-matrices are 4-by-8 and 8-by-4, respectively, while the int32 C-matrix is 4-by-4. This represents a quad-widening of the arithmetic values, but only a double-widening of the data width due to the destination matrix being smaller than the input matrices.

The products of the 8-bit signed or unsigned integer inputs are calculated exactly and are added to the 32-bit integer accumulators, with wraparound if the 32-bit destination overflows.

These instructions are encoded similarly to other vector instructions, but within the *custom-2* major opcode, with `funct3`=2 (OPMVV), and with `funct6` as specified below.

```
# Quad-widening unsigned-integer matrix-multiply-add, overwrite addend
sf.vqmaccu.4x8x4  vd, vs1, vs2  # funct6=111100

# Quad-widening signed-integer matrix-multiply-add, overwrite addend
sf.vqmacc.4x8x4   vd, vs1, vs2  # funct6=111101

# Quad-widening unsigned-signed-integer matrix-multiply-add, overwrite addend
sf.vqmaccus.4x8x4 vd, vs1, vs2  # funct6=111110

# Quad-widening signed-unsigned-integer matrix-multiply-add, overwrite addend
sf.vqmaccsu.4x8x4 vd, vs1, vs2  # funct6=111111
```

The above instructions perform `vl`/32 matrix operations,

```
C[j] += A * B[j], for j in [0, vl/32)
```

where

- The 4-by-8 (u)int8 matrix `A` resides in row-major layout in bytes `[0, 32)` of `vs1`.
- The 8-by-4 (u)int8 matrix `B[j]` resides in row-major layout in bytes `[j*32, (j+1)*32)` of `vs2`.
- The 4-by-4 int32 matrix `C[j]` resides in row-major layout in bytes `[j*64, (j+1)*64)` of `vd`.

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

The instructions are reserved if `vl` is not divisible by 32, or if SEW≠8, or if masked. `vd` must not overlap `vs1`.

1.2 Xsfvqmaccdod Extension: Int8 Matrix Multiplication Instructions (2-by-8 and 8-by-2)

These instructions are signed or unsigned integer matrix-multiply-add instructions that widen 8-bit integer input values to 32-bits. To understand the functionality of these instructions, consider the arithmetic below:

```
# Dest (C)   Input (A)  *  Input (B)
  int32 +=       uint8  *      uint8
  int32 +=       uint8  *       int8
  int32 +=        int8  *      uint8
  int32 +=        int8  *       int8
```

Here, the (u)int8 A- and B-matrices are 2-by-8 and 8-by-2 respectively, while the int32 C-matrix is 2-by-2. This represents a quad-widening of the arithmetic values, but only a double-widening of the data width due to the destination matrix being smaller than the input matrices.

The products of the 8-bit signed or unsigned integer inputs are calculated exactly, and are added to the 32-bit integer accumulators, with wraparound if the 32-bit destination overflows.

These instructions are encoded similarly to other vector instructions, but within the *custom-2* major opcode, with `funct3`=2 (OPMVV), and with `funct6` as specified below.

```
# Quad-widening unsigned-integer matrix-multiply-add, overwrite addend
sf.vqmaccu.2x8x2  vd, vs1, vs2  # funct6=101100

# Quad-widening signed-integer matrix-multiply-add, overwrite addend
sf.vqmacc.2x8x2   vd, vs1, vs2  # funct6=101101

# Quad-widening unsigned-signed-integer matrix-multiply-add, overwrite addend
sf.vqmaccus.2x8x2 vd, vs1, vs2  # funct6=101110

# Quad-widening signed-unsigned-integer matrix-multiply-add, overwrite addend
```

```
sf.vqmaccsu.2x8x2 vd, vs1, vs2  # funct6=101111
```

The above instructions perform `vl`/16 matrix operations,

```
C[j] += A * B[j], for j in [0, vl/16)
```

where

- The 2-by-8 (u)int8 matrix `A` resides in row-major layout in bytes `[0, 16)` of `vs1`.
- The 8-by-2 (u)int8 matrix `B[j]` resides in row-major layout in bytes `[j*16, (j+1)*16)` of `vs2`.
- The 2-by-2 int32 matrix `C[j]` resides in row-major layout in bytes `[j*16, (j+1)*16)` of `vd`.

`vs1` has EMUL=1; `vs2` and `vd` have EMUL=LMUL.

The instructions are reserved if `vl` is not divisible by 16, if SEW≠8, or if masked. `vd` must not overlap `vs1`.

1.3 C Intrinsics

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

```
vint32m1_t __riscv_sf_vqmacc_4x8x4_i32m1(vint32m1_t vd, vint8m1_t vs1, vint8mf2_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmacc_4x8x4_i32m2(vint32m2_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmacc_4x8x4_i32m4(vint32m4_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmacc_4x8x4_i32m8(vint32m8_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmacc_4x8x4(vint32m1_t vd, vint8m1_t vs1, vint8mf2_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmacc_4x8x4(vint32m2_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmacc_4x8x4(vint32m4_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmacc_4x8x4(vint32m8_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmacc_4x8x4_i32m1_tu(vint32m1_t vd, vint8m1_t vs1, vint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmacc_4x8x4_i32m2_tu(vint32m2_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmacc_4x8x4_i32m4_tu(vint32m4_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmacc_4x8x4_i32m8_tu(vint32m8_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmacc_4x8x4_tu(vint32m1_t vd, vint8m1_t vs1, vint8mf2_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmacc_4x8x4_tu(vint32m2_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmacc_4x8x4_tu(vint32m4_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmacc_4x8x4_tu(vint32m8_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t vl)



vint32m1_t __riscv_sf_vqmaccsu_4x8x4_i32m1(vint32m1_t vd, vint8m1_t vs1, vuint8mf2_t vs2,
size_t vl)
```

```
vint32m2_t __riscv_sf_vqmaccsu_4x8x4_i32m2(vint32m2_t vd, vint8m1_t vs1, vuint8m1_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmaccsu_4x8x4_i32m4(vint32m4_t vd, vint8m1_t vs1, vuint8m2_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmaccsu_4x8x4_i32m8(vint32m8_t vd, vint8m1_t vs1, vuint8m4_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmaccsu_4x8x4(vint32m1_t vd, vint8m1_t vs1, vuint8mf2_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccsu_4x8x4(vint32m2_t vd, vint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccsu_4x8x4(vint32m4_t vd, vint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccsu_4x8x4(vint32m8_t vd, vint8m1_t vs1, vuint8m4_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmaccsu_4x8x4_i32m1_tu(vint32m1_t vd, vint8m1_t vs1, vuint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccsu_4x8x4_i32m2_tu(vint32m2_t vd, vint8m1_t vs1, vuint8m1_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccsu_4x8x4_i32m4_tu(vint32m4_t vd, vint8m1_t vs1, vuint8m2_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccsu_4x8x4_i32m8_tu(vint32m8_t vd, vint8m1_t vs1, vuint8m4_t vs2,
size_t vl)

vint32m1_t __riscv_sf_vqmaccsu_4x8x4_tu(vint32m1_t vd, vint8m1_t vs1, vuint8mf2_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmaccsu_4x8x4_tu(vint32m2_t vd, vint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccsu_4x8x4_tu(vint32m4_t vd, vint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccsu_4x8x4_tu(vint32m8_t vd, vint8m1_t vs1, vuint8m4_t vs2, size_t vl)



vint32m1_t __riscv_sf_vqmaccu_4x8x4_i32m1(vint32m1_t vd, vuint8m1_t vs1, vuint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccu_4x8x4_i32m2(vint32m2_t vd, vuint8m1_t vs1, vuint8m1_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmaccu_4x8x4_i32m4(vint32m4_t vd, vuint8m1_t vs1, vuint8m2_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmaccu_4x8x4_i32m8(vint32m8_t vd, vuint8m1_t vs1, vuint8m4_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmaccu_4x8x4(vint32m1_t vd, vuint8m1_t vs1, vuint8mf2_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccu_4x8x4(vint32m2_t vd, vuint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccu_4x8x4(vint32m4_t vd, vuint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccu_4x8x4(vint32m8_t vd, vuint8m1_t vs1, vuint8m4_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmaccu_4x8x4_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1, vuint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccu_4x8x4_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1, vuint8m1_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccu_4x8x4_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1, vuint8m2_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccu_4x8x4_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1, vuint8m4_t vs2,
size_t vl)

vint32m1_t __riscv_sf_vqmaccu_4x8x4_tu(vint32m1_t vd, vuint8m1_t vs1, vuint8mf2_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmaccu_4x8x4_tu(vint32m2_t vd, vuint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccu_4x8x4_tu(vint32m4_t vd, vuint8m1_t vs1, vuint8m2_t vs2, size_t vl)
```

```
vint32m8_t __riscv_sf_vqmaccu_4x8x4_tu(vint32m8_t vd, vuint8m1_t vs1, vuint8m4_t vs2, size_t vl)
```

```
vint32m1_t __riscv_sf_vqmaccus_4x8x4_i32m1(vint32m1_t vd, vuint8m1_t vs1, vint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccus_4x8x4_i32m2(vint32m2_t vd, vuint8m1_t vs1, vint8m1_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmaccus_4x8x4_i32m4(vint32m4_t vd, vuint8m1_t vs1, vint8m2_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmaccus_4x8x4_i32m8(vint32m8_t vd, vuint8m1_t vs1, vint8m4_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmaccus_4x8x4(vint32m1_t vd, vuint8m1_t vs1, vint8mf2_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccus_4x8x4(vint32m2_t vd, vuint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccus_4x8x4(vint32m4_t vd, vuint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccus_4x8x4(vint32m8_t vd, vuint8m1_t vs1, vint8m4_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmaccus_4x8x4_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1, vint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccus_4x8x4_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1, vint8m1_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccus_4x8x4_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1, vint8m2_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccus_4x8x4_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1, vint8m4_t vs2,
size_t vl)

vint32m1_t __riscv_sf_vqmaccus_4x8x4_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1, vint8mf2_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccus_4x8x4_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1, vint8m1_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccus_4x8x4_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1, vint8m2_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccus_4x8x4_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1, vint8m4_t vs2,
size_t vl)
```

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

```
vint32m1_t __riscv_sf_vqmacc_2x8x2_i32m1(vint32m1_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmacc_2x8x2_i32m2(vint32m2_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmacc_2x8x2_i32m4(vint32m4_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmacc_2x8x2_i32m8(vint32m8_t vd, vint8m1_t vs1, vint8m8_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmacc_2x8x2(vint32m1_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmacc_2x8x2(vint32m2_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmacc_2x8x2(vint32m4_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmacc_2x8x2(vint32m8_t vd, vint8m1_t vs1, vint8m8_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmacc_2x8x2_i32m1_tu(vint32m1_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmacc_2x8x2_i32m2_tu(vint32m2_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmacc_2x8x2_i32m4_tu(vint32m4_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmacc_2x8x2_i32m8_tu(vint32m8_t vd, vint8m1_t vs1, vint8m8_t vs2, size_t
```

```
vl)

vint32m1_t __riscv_sf_vqmacc_2x8x2_tu(vint32m1_t vd, vint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmacc_2x8x2_tu(vint32m2_t vd, vint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmacc_2x8x2_tu(vint32m4_t vd, vint8m1_t vs1, vint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmacc_2x8x2_tu(vint32m8_t vd, vint8m1_t vs1, vint8m8_t vs2, size_t vl)


vint32m1_t __riscv_sf_vqmaccsu_2x8x2_i32m1(vint32m1_t vd, vint8m1_t vs1, vuint8m1_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmaccsu_2x8x2_i32m2(vint32m2_t vd, vint8m1_t vs1, vuint8m2_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmaccsu_2x8x2_i32m4(vint32m4_t vd, vint8m1_t vs1, vuint8m4_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmaccsu_2x8x2_i32m8(vint32m8_t vd, vint8m1_t vs1, vuint8m8_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmaccsu_2x8x2(vint32m1_t vd, vint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccsu_2x8x2(vint32m2_t vd, vint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccsu_2x8x2(vint32m4_t vd, vint8m1_t vs1, vuint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccsu_2x8x2(vint32m8_t vd, vint8m1_t vs1, vuint8m8_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmaccsu_2x8x2_i32m1_tu(vint32m1_t vd, vint8m1_t vs1, vuint8m1_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccsu_2x8x2_i32m2_tu(vint32m2_t vd, vint8m1_t vs1, vuint8m2_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccsu_2x8x2_i32m4_tu(vint32m4_t vd, vint8m1_t vs1, vuint8m4_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccsu_2x8x2_i32m8_tu(vint32m8_t vd, vint8m1_t vs1, vuint8m8_t vs2,
size_t vl)

vint32m1_t __riscv_sf_vqmaccsu_2x8x2_tu(vint32m1_t vd, vint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccsu_2x8x2_tu(vint32m2_t vd, vint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccsu_2x8x2_tu(vint32m4_t vd, vint8m1_t vs1, vuint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccsu_2x8x2_tu(vint32m8_t vd, vint8m1_t vs1, vuint8m8_t vs2, size_t vl)


vint32m1_t __riscv_sf_vqmaccu_2x8x2_i32m1(vint32m1_t vd, vuint8m1_t vs1, vuint8m1_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmaccu_2x8x2_i32m2(vint32m2_t vd, vuint8m1_t vs1, vuint8m2_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmaccu_2x8x2_i32m4(vint32m4_t vd, vuint8m1_t vs1, vuint8m4_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmaccu_2x8x2_i32m8(vint32m8_t vd, vuint8m1_t vs1, vuint8m8_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmaccu_2x8x2(vint32m1_t vd, vuint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccu_2x8x2(vint32m2_t vd, vuint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccu_2x8x2(vint32m4_t vd, vuint8m1_t vs1, vuint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccu_2x8x2(vint32m8_t vd, vuint8m1_t vs1, vuint8m8_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmaccu_2x8x2_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1, vuint8m1_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccu_2x8x2_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1, vuint8m2_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccu_2x8x2_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1, vuint8m4_t vs2,
```

```
size_t vl)
vint32m8_t __riscv_sf_vqmaccu_2x8x2_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1, vuint8m8_t vs2,
size_t vl)

vint32m1_t __riscv_sf_vqmaccu_2x8x2_tu(vint32m1_t vd, vuint8m1_t vs1, vuint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccu_2x8x2_tu(vint32m2_t vd, vuint8m1_t vs1, vuint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccu_2x8x2_tu(vint32m4_t vd, vuint8m1_t vs1, vuint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccu_2x8x2_tu(vint32m8_t vd, vuint8m1_t vs1, vuint8m8_t vs2, size_t vl)


vint32m1_t __riscv_sf_vqmaccus_2x8x2_i32m1(vint32m1_t vd, vuint8m1_t vs1, vint8m1_t vs2, size_t
vl)
vint32m2_t __riscv_sf_vqmaccus_2x8x2_i32m2(vint32m2_t vd, vuint8m1_t vs1, vint8m2_t vs2, size_t
vl)
vint32m4_t __riscv_sf_vqmaccus_2x8x2_i32m4(vint32m4_t vd, vuint8m1_t vs1, vint8m4_t vs2, size_t
vl)
vint32m8_t __riscv_sf_vqmaccus_2x8x2_i32m8(vint32m8_t vd, vuint8m1_t vs1, vint8m8_t vs2, size_t
vl)

vint32m1_t __riscv_sf_vqmaccus_2x8x2(vint32m1_t vd, vuint8m1_t vs1, vint8m1_t vs2, size_t vl)
vint32m2_t __riscv_sf_vqmaccus_2x8x2(vint32m2_t vd, vuint8m1_t vs1, vint8m2_t vs2, size_t vl)
vint32m4_t __riscv_sf_vqmaccus_2x8x2(vint32m4_t vd, vuint8m1_t vs1, vint8m4_t vs2, size_t vl)
vint32m8_t __riscv_sf_vqmaccus_2x8x2(vint32m8_t vd, vuint8m1_t vs1, vint8m8_t vs2, size_t vl)

vint32m1_t __riscv_sf_vqmaccus_2x8x2_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1, vint8m1_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccus_2x8x2_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1, vint8m2_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccus_2x8x2_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1, vint8m4_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccus_2x8x2_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1, vint8m8_t vs2,
size_t vl)

vint32m1_t __riscv_sf_vqmaccus_2x8x2_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1, vint8m1_t vs2,
size_t vl)
vint32m2_t __riscv_sf_vqmaccus_2x8x2_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1, vint8m2_t vs2,
size_t vl)
vint32m4_t __riscv_sf_vqmaccus_2x8x2_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1, vint8m4_t vs2,
size_t vl)
vint32m8_t __riscv_sf_vqmaccus_2x8x2_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1, vint8m8_t vs2,
size_t vl)
```