The Scalable Matrix Extension (SME), for Armv9-A

Known issues in Issue B.a

Non-Confidential

Issue 00
109353_B.a_00_en

The Scalable Matrix Extension (SME), for Armv9-A

Known issues in Issue B.a

Release information

Document history

Issue	Date	Confidentiality	Change
B.a-00	30 September 2023	Non-Confidential	Known Issues in The Scalable Matrix Extension (SME), for Armv9-A (DDI 0616), as of 22 September 2023

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction

1.1 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` <u>`underline`</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
Note	An important piece of information that needs your attention.

Convention	Use
Tip	A useful tip that might make it easier, better or faster to perform a task.
Remember	A reminder of something important that relates to the information you are reading.

1.2 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® Architecture Reference Manual Supplement, The Scalable Matrix Extension (SME), for Armv9-A	DDI 0616	Non-Confidential

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com

1.3 Other information

See the Arm website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.
- Arm® Glossary.

2. Known issues

This document records known issues in the Arm Architecture Reference Manual Supplement, The Scalable Matrix Extension (SME), for Armv9-A (DDI 0616), Issue B.a.

Key

- C = Clarification.
- D = Defect.
- R = Relaxation.
- E = Enhancement.

2.1 C1402: SME

In sections D1.1.27 (FCLAMP) and D1.2.5 (FCLAMP), the text that reads:

> If at least one element value contributing to a result is numeric and the others are either numeric or a quiet NaN, then the result is the numeric value.

is changed to read:

> Regardless of the value of FPCR.AH, the behavior is as follows for each minimum number and maximum number operation:
>
> - Negative zero compares less than positive zero.
> - If one input is numeric and the other is a quiet NaN, the result is the numeric value.
> - When FPCR.DN is 0, if either value is a signaling NaN or if both values are NaNs, the result is a quiet NaN.
> - When FPCR.DN is 1, if either value is a signaling NaN or if both values are NaNs, the result is Default NaN.

2.2 D1423: SME

In section E3.1.6 (SMIDR_EL1, Streaming Mode Identification Register), the following new fields are added:

> Affinity2, bits [51:32]
>
> The most significant 20 bits of the SMCU affinity for this PE, to be used in conjunction with SMIDR_EL1.Affinity.
>
> SH, bits [14:13]

> Indicates whether the implementation of Streaming SVE mode in this PE is shared with other PEs.

SH	Meaning
`0b00`	Refer to SMIDR_EL1.Affinity.
`0b01`	Reserved.
`0b10`	The implementation of Streaming SVE mode is not shared with other PEs.
`0b11`	The implementation of Streaming SVE mode is shared with other PEs.

In the same section, the field 'Affinity, bits [11:0]' that reads:

> The SMCU affinity of the accessing PE.
>
> - A value of zero indicates that the PE's implementation of Streaming SVE mode is not shared with other PEs.
> - Otherwise, the value identifies which SMCU is associated with this PE. The Affinity value associated with each SMCU is unique within the system as a whole.

is changed to read:

> The least significant 12 bits of the SMCU affinity for this PE.
>
> - If SMIDR_EL1.SH is `0b00` and this field is zero, then the implementation of Streaming SVE mode is not shared with other PEs.
> - If SMIDR_EL1.SH is `0b00` and this field is not zero, then the implementation of Streaming SVE mode is shared with other PEs.
> - Otherwise, SMIDR_EL1.SH indicates whether the implementation of Streaming SVE mode is shared with other PEs.
>
> If the implementation of Streaming SVE mode is shared, then the concatenated value SMIDR_EL1.{Affinity2,Affinity} identifies which shared SMCU is associated with the PE. The 32-bit value associated with each SMCU is unique within the system as a whole.

Additionally, in section C1.2.4 (Streaming execution priority for shared implementations), the rules that read:

> R_{WPVQK}
>
> All PEs in a Priority domain have the same value of SMIDR_EL1.Affinity.
>
> R_{CVLSF}
>
> PEs in differing Priority domains have different values of SMIDR_EL1.Affinity.

are changed to read:

> R_{WPVQK}
>
> All PEs in a Priority domain have the same value of SMIDR_EL1.Affinity and SMIDR_EL1.Affinity2.

R_{CVLSF}

PEs in differing Priority domains have different values of SMIDR_EL1.Affinity or SMIDR_EL1.Affinity2.

2.3 D1441: SME

In section D1.1.104 (LUTI2 (two registers)), the Operation pseudocode that reads:

```
for r = 0 to nreg-1
    integer base = (segment * nreg + r) * elements;
    bits(VL) result = Z[d, VL];
    for e = 0 to elements-1
        integer index = UInt(Elem[indexes, base+e, isize]);
        Elem[result, e, esize] = Elem[table, index, 32]<esize-1:0>;
    Z[d, VL] = result;
```

is corrected to remove the spurious read of Z[d]:

```
for r = 0 to nreg-1
    integer base = (segment * nreg + r) * elements;
    bits(VL) result;
    for e = 0 to elements-1
        integer index = UInt(Elem[indexes, base+e, isize]);
        Elem[result, e, esize] = Elem[table, index, 32]<esize-1:0>;
    Z[d, VL] = result;
```

The same change is applied in the following sections:

- D1.1.105 (LUTI2 (four registers)).
- D1.1.106 (LUTI2 (single)).
- D1.1.107 (LUTI4 (two registers)).
- D1.1.108 (LUTI4 (four registers)).
- D1.1.109 (LUTI4 (single)).

2.4 D1458: SME

In sections D1.1.102 (LDR (vector)) and D1.1.218 (STR (vector)), the Operation pseudocode that reads:

```
integer moffs = offset * dim;
...
boolean aligned = IsAligned(base + offset, 16);
if !aligned && AlignmentEnforced() then
    AArch64.Abort(base + moffs, AlignmentFault(accdesc));
```

is corrected to read:

```
integer moffs = offset * dim;
...
boolean aligned = IsAligned(base + moffs, 16);
if !aligned && AlignmentEnforced() then
    AArch64.Abort(base + moffs, AlignmentFault(accdesc));
```

2.5 C1588: SME

In section C1.2.1 (Exception priorities), the following rule is added after R_{ZZBRC}:

> When an SVE MOVPRFX instruction predictably prefixes an SVE instruction that is illegal due to the current value of PSTATE.SM or PSTATE.ZA, then execution of either of the instructions generates a synchronous SME exception with ESR_ELx.EC value `0x1D` and an ISS code that is not `0x0000000`, consistent with the behaviors defined by rule R_{RWVTR} in subsection SVE MOVPRFX exception entry behavior of Arm Architecture Reference Manual for A-profile architecture [1].

2.6 C1603: SME

The following SME instruction description sections:

- D1.1.102 (LDR (vector)).
- D1.1.103 (LDR (ZT0)).
- D1.1.130 (MOVT (ZT0 to scalar)).
- D1.1.131 (MOVT (scalar to ZT0)).
- D1.1.218 (STR (vector)).
- D1.1.219 (STR (ZT0)).
- D1.1.291 (ZERO (tile)).
- D1.1.292 (ZERO (ZT0)).

are changed to read:

- D1.1.102 “LDR (array vector)”.
- D1.1.103 “LDR (table)”.
- D1.1.130 “MOVT (table to scalar)”.
- D1.1.131 “MOVT (scalar to table)”.
- D1.1.218 “STR (array vector)”.
- D1.1.219 “STR (table)”.
- D1.1.291 “ZERO (tiles)”.

- D1.1.292 “ZERO (table)”.

The changes to the SME qualified instruction titles are applied throughout the supplement.

2.7 R1610: SME

In section D1.1.121 (MOVA (tile to vector, four registers)), in the ‘64-bit’ encoding variant, the decode pseudocode that reads:

```
if !HaveSME2() then UNDEFINED;
...
```

is updated to read:

```
if !HaveSME2() then UNDEFINED;
if MaxImplementedSVL() < 256 then UNDEFINED;
...
```

Similarly, in section D1.1.282 (UZP (two registers)), in the ‘128-bit’ encoding variant, the decode pseudocode that reads:

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 128;
...
```

is updated to read:

```
if !HaveSME2() then UNDEFINED;
if MaxImplementedSVL() < 512 then UNDEFINED;
constant integer esize = 128;
...
```

Equivalent changes are made in the following sections:

- D1.1.126 (MOVA (vector to tile, four registers)), when esize == 64.
- D1.1.281 (UZP (four registers)), when esize >= 64.
- D1.1.293 (ZIP (four registers)), when esize >= 64.
- D1.1.294 (ZIP (two registers)), when esize == 128.