Trusted Firmware-M

FP support in TF-M Update

2022 March

Feder Liang
Arm

FP support in TF-M (Update)

- Armv8.0 FP support (IPC, SFN)
- Armv8.1 FP support (IPC, SFN)
- M-Profile Vector Extension (MVE) support vs. FP support

FP Context

FP context is shared between S and NS.

Name	Description	Security State Banked
CPACR	Coprocessor Access Control Register	Yes
CPPWR	Coprocessor Power Control Register	No
FP Register Bank	FP caller save registers (S0–S15) FP callee save registers (S16–S31)	No
FPSCR	Floating-point Status and Control Register	No
FPCCR	Floating Point Context Control Register	Partial
	• LSPEN Enable/Disable lazy stacking	No
FPCAR	Floating Point Context Address Register	Yes
FPDSCR	Floating Point Default Status Control Register	Yes
MVFR0, 1, 2	Media and FP Feature Register 0, 1, 2	No

FP context to be protected:

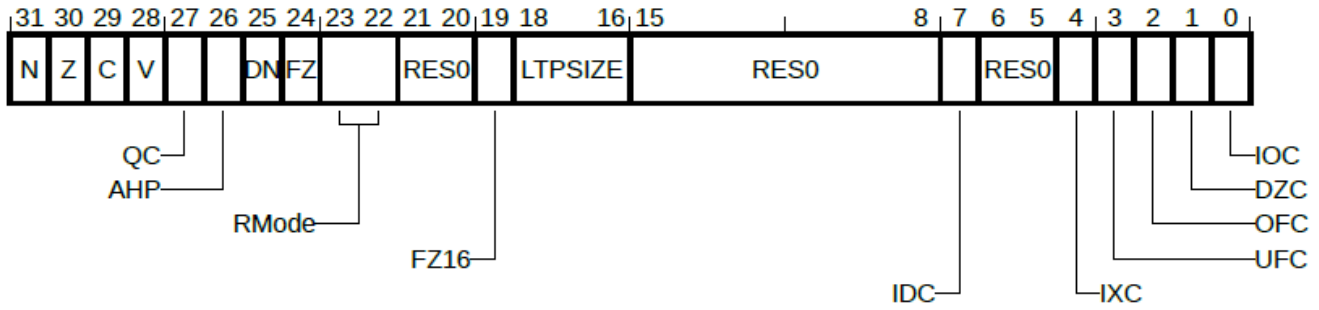
- FP registers (S0-S31)
- FPSCR

FPSCR

Floating-point Status and Control Register

- Providing status information about the floating-point operation results.
- Defining some of the floating-point operation behaviors.
- The vector element size used when applying low-overhead-loop tail predication to vector instructions.
- The exception bits can be used by software to detect abnormalities in floating point operations

The FPSCR bit assignments are:



Armv8.0 FP support for IPC Model

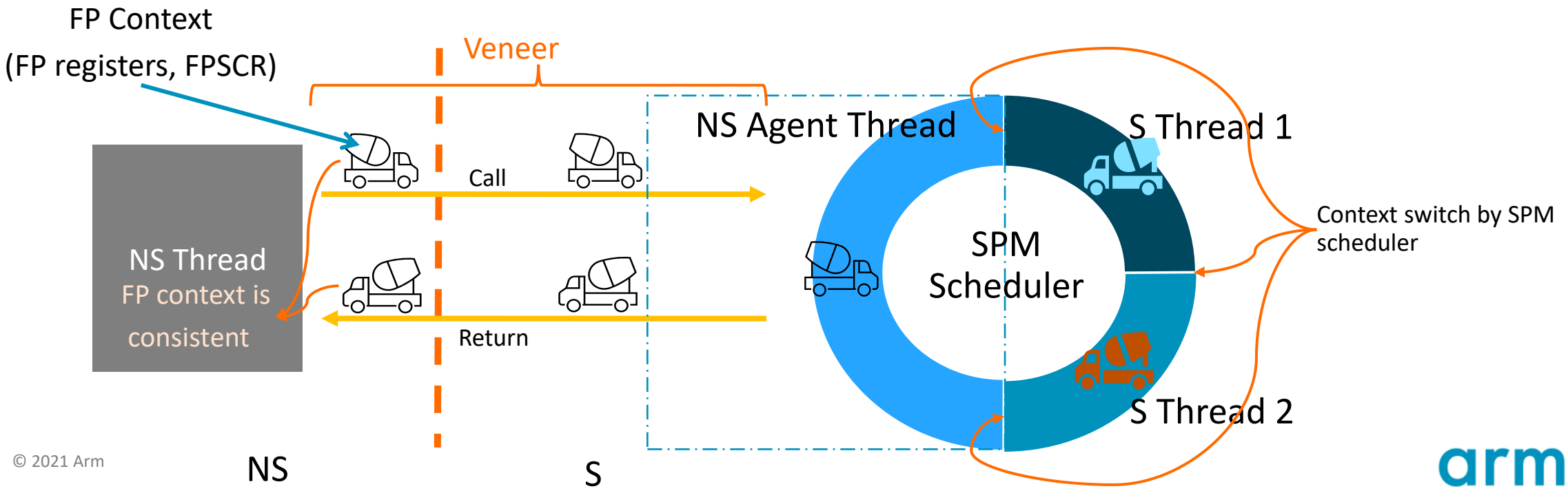
For processor support armv8-m.main

Items	Details	
Submit	Release 1.5.0	5519438
Scope	SPE only	SPE and **NSPE**
FP context protection mechanism	FP context saved in secure partition's stack.	
	FP context is saved (also invalidated) and restore automatically by cortex-m processor hardware mechanism during exception entry and exception return.	
	Prevent non-secure from modifying FPU's power setting (CPPWR).	
Toolchain	GNU Arm embedded toolchain	GNU Arm embedded toolchain
ABI	soft, **softfp**, hard	soft, hard
NS FPU usage	Disable non-secure access to Floating-point Unit (FPU).	**Permit** non-secure access to FPU.
Lazy stacking	Enable/Disable lazy stacking in SPE only.	Lazy stacking can only be enabled or disabled for **whole system** from **SPE**(LSPENS).

Armv8.0 FP support for IPC model in TF-M

Lazy stacking disabled

- Veneer code is written as assembly code in TF-M.
- NS FP context is saved and restored on NS agent thread's stack. FPSCR is **consistent** for NS.
- Secure FP context is saved (also invalidated) and restored on SP's stack before context switch to NS agent thread. FPSCR is also **consistent** for each SP.



Armv8.0 FP support for SFN Model (Under development)

For processor support armv8-m.main, isolation level 1

Items	Details
Scope	SPE and NSPE
FP context protection mechanism	Veneer implemented in assembly code.
	Secure FP context are invalidated before function return to NS if secure FP context is active.
	FPSCR is saved on NS agent thread's stack before secure function call and is restored before function return to NS.
	Prevent non-secure from modifying FPU's power setting (CPPWR).
Toolchain	GNU Arm embedded toolchain
ABI	soft, hard
NS FPU usage	Permit non-secure access to FPU.
Lazy stacking	Lazy stacking can only be enabled or disabled for whole system from SPE (LSPENS).

Armv8.0 FP support for SFN model (Demo)

```
__tfm_psa_secure_gateway_attributes__
psa_status_t tfm_psa_call_veneer(psa_handle_t handle, uint32_t
ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)
{
    __ASM volatile(
        ".syntax unified                                    \n"
        "   push   {r2, r3}                                 \n"
        "   ldr    r2, [sp, #8]                             \n"
        "   ldr    r3, ="M2S(STACK_SEAL_PATTERN)"           \n"
        "   cmp    r2, r3                                   \n"
        "   bne    reent_panic4                             \n"
        "   pop    {r2, r3}                                 \n"
        "   mov    r12, r3                                  \n"
        "   mrs    r3, control                              \n"
        "   push   {r2, r3}                                 \n"
        "   mov    r3, r12                                  \n"
        "   push   {lr}                                     \n"

        "   vmrs   r4, fpscr                                \n"
        "   push   {r4}                                     \n"

        "   bl     psa_call_pack_sfn                        \n"

        "   pop    {r4}                                     \n"
        "   vmsr   fpscr, r4                                \n"

        "   mrs    r4, control                              \n"
        "   tst.w  r4, #8                                   \n"
        "   beq    no_sfpa2                                 \n"

        "   eor    r2, r2, r2                               \n"
        "   vmov   d0, r2, r2                               \n"
        "   vmov   d1, r2, r2                               \n"
        "   vmov   d2, r2, r2                               \n"
        "   vmov   d3, r2, r2                               \n"
        "   vmov   d4, r2, r2                               \n"
        "   vmov   d5, r2, r2                               \n"
        "   vmov   d6, r2, r2                               \n"
        "   vmov   d7, r2, r2                               \n"
        "no_sfpa2:                                          \n"

        "   pop    {r2, r3}                                 \n"
        "   mov    lr, r3                                   \n"
        "   pop    {r2, r3}                                 \n"
        "   msr    control, r3                              \n"
        "   bxns   lr                                       \n"
        "reent_panic4:                                      \n"
        "   svc    "M2S(TFM_SVC_PSA_PANIC)"                 \n"
        "   b      .                                        \n"
    );
}
```

Save FPSCR(NS)

Restore FPSCR(NS)

Check SFPA (Secure Floating-point active.)

Clear FP caller registers

Armv8.1 Features for FP

For processor support armv8.1-m.main

- FPCXT
  - Usage for function call.
  - Avoid corrupting of providing inconsistent of FPSCR between S and NS.

- VSCCLRM
  - Avoid trigger the creation of inadvertent FP context during invalidation of S FP context before return to NS.
  - Because it doesn’t clear any registers if there isn’t a secure context active (as indicated by CONTROL_S.SFPA).
  - Side effect of create FP context inadvertently
    - Because this FP context has to be saved and restored for every context switch, it wastes time, stack space, power etc. for the rest of the lifetime of that thread.

FPCXT in Armv8.1

- FPSCR

Bit	Field	Descriptions
31	N	Negative condition flag.
30	Z	Zero condition flag.
29	C	Carry condition flag.
28	V	Overflow condition flag.
27	QC	Cumulative saturation bit.
26	AHP	Alternative half-precision control bit.
25	DN	Default NaN mode control bit.
24	FZ	Flush-to-zero mode control for single and double precision Floating-point.
23:22	RMode	Rounding mode control field.
21:20		Reserved.
19	FZ16	Flush-to-zero mode control bit on half-precision data-processing instructions.
18:16	LTPSIZE	The vector element size used when applying low-overhead-loop tail predication to vector instructions.
15:08		Reserved.
7	IDC	Input Denormal cumulative exception bit.
6:5		Reserved.
4	IXC	Inexact cumulative exception bit.
3	UFC	Underflow cumulative exception bit.
2	OFC	Overflow cumulative exception bit.
1	DZC	Divide by Zero cumulative exception bit.
0	IOC	Invalid Operation cumulative exception bit.

- FPCXT Floating-point context payload
  - Introduced in V8.1 to provide consistent FPSCR for S or NS both.
  - Save/restore during security changes, avoid inconsistent FPSCR occurs.

Bit	Field	Descriptions
31	SFPA	Secure Floating-point active. CONTROL.SFPA
30		Reserved.
29		Reserved.
28		Reserved.
27	QC	Cumulative saturation bit.
26	AHP	Alternative half-precision control bit.
25	DN	Default NaN mode control bit.
24	FZ	Flush-to-zero mode control for single and double precision Floating-point.
23:22	RMode	Rounding mode control field.
21:20		Reserved.
19	FZ16	Flush-to-zero mode control bit on half-precision data-processing instructions.
18:16	LTPSIZE	The vector element size used when applying low-overhead-loop tail predication to vector instructions.
15:08		Reserved.
7	IDC	Input Denormal cumulative exception bit.
6:5		Reserved.
4	IXC	Inexact cumulative exception bit.
3	UFC	Underflow cumulative exception bit.
2	OFC	Overflow cumulative exception bit.
1	DZC	Divide by Zero cumulative exception bit.
0	IOC	Invalid Operation cumulative exception bit.

Same

Armv8.1 FP support in TF-M for IPC model

For processor support armv8.1-m.main

Items	Details
Scope	SPE and NSPE
FP context protection mechanism (Same as Armv8.0 FP support)	FP context saved in secure partition's stack.
	FP context is saved (also invalidated) and restored automatically by cortex-m processor hardware mechanism during exception entry and exception return.
	Prevent non-secure from modifying FPU's power setting (CPPWR).
Toolchain	GNU Arm embedded toolchain: add **__ARM_ARCH_8_1M_MAIN__** manually.
ABI	soft, hard
NS FPU usage	Permit non-secure access to FPU.
Lazy stacking	Lazy stacking can only be enabled or disabled for whole system from SPE (LSPENS).

FPCXT and VSCCLRM are not used

Armv8.1 FP support for SFN model (Under development)

For processor support armv8.1-m.main

Items	Details
Scope	SPE and NSPE
FP context protection mechanism	Veneer implemented in assembly code
	Secure FP context are invalidated before function return to NS if secure FP context active: By **VSCCLRM**
	FPSCR is saved before secure function call: **vstr FPCXTNS, [sp, #-4]!** And is restored before function return to NS: **vldr FPCXTNS, [sp], #4**
	Prevent non-secure from modifying FPU's power setting (CPPWR).
Toolchain	GNU Arm embedded toolchain
ABI	soft, hard
NS FPU usage	Permit non-secure access to FPU.
Lazy stacking	Lazy stacking can only be enabled or disabled for whole system from SPE (LSPENS).

Armv8.1 FP support for SFN model (Demo)

```
psa_status_t tfm_psa_call_veneer(psa_handle_t h, uint32_t ctrl, const psa_invec *in_vec, psa_outvec *out_vec)
{
    __ASM volatile(
      ".syntax unified                    \n"
      "   push   {lr}                     \n"
      "   vstr   FPCXTNS, [sp, #-4]!      \n"
      "   bl     psa_call_pack_sfn        \n" /* ABI to psa framework */
      "   vscclrm {s0-s15, vpr}           \n"
      "   vldr   FPCXTNS, [sp], #4        \n"
      "   pop    {r3}                     \n"
      "   mov    lr, r3                   \n"
      "   bxns   lr                       \n"
    );
}
```

Save FPCXTNS (FPSCR NS)

Clear FP caller registers if secure context active (as indicated by CONTROL_S.SFPA)

Restore FPCXTNS (FPSCR NS)

MVE support vs. FP support

For processor support armv8.1-m.main

- M-Profile Vector Extension (MVE) is an optional vector architectural extension introduced as part of the ARMv8.1-M architecture, for the Arm Cortex-M processor series.
- MVE delivers a significant performance uplift for machine learning and digital signal processing applications for small, embedded devices.
- MVE reuses FP registers, enabling MVE is same as FP does.
- MVE working routine are same as FP mechanism.
  - Such as CONTROL.FPCA, CONTROL.SFPA, FPCCR.LSPACT, lazy stacking, etc.
  - Besides FP registers and FPSCR, VPR register are also pushed to stack by hardware during exception entry.
- MVE is perfectly designed to exploit all existing security mechanisms already designed for FP.

arm

Live Q & A

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה