TF-M Dual-cpu
NS Mailbox Improvement

Enhance integration with NS environment

David Hu
2020 Nov
Agenda

• TF-M dual-cpu NS mailbox enhancement
  • Simplify NS RTOS port

• Enhance NS mailbox working model
  • A new working model to support NS applications isolation
  • Refine working model configuration
arm

NS mailbox enhancement
Various RTOS modules distributed in TF-M

- Semaphores inside PSA Client API implementation
  - A dedicated API set outside NS mailbox
    ```c
    uint32_t psa_framework_version(void)
    {
      if (tfm_ns_multi_core_lock_acquire() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
      }
      /* mailbox handling */
      if (tfm_ns_multi_core_lock_release() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
      }
    }
    ```
  - Platform specific NS mailbox .c

- Part of thread mgmt. placed in platform specific driver
  - RTOS specific thread mgmt.
  - Platform independent

- Improve goals:
  - Sort out mailbox interface
  - Improve dependencies on RTOS

Improve goals:
- Sort out mailbox interface
- Improve dependencies on RTOS
Simplify NS mailbox API

- A single NS mailbox API `tfm_ns_mailbox_client_call()`
  - Combine various NS mailbox APIs
  - Avoid exporting NS mailbox internal variables

```c
uint32_t psa_framework_version(void) {
    mailbox_msg_handle_t handle;
    ...
    if (tfm_ns_multi_core_lock_acquire() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
    }
    handle = tfm_ns_mailbox_tx_client_req(...);
    ...
    mailbox_wait_reply(handle);
    ret = tfm_ns_mailbox_rx_client_reply(handle, ...);
    ...
    if (tfm_ns_multi_core_lock_release() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
    }
    return version;
}
```
Re-organize NS mailbox dependencies on RTOS

Decouple ROTS specific impl. from Platform and common NS mailbox

- Define NS mailbox RTOS API
  - tfm_ns_mailbox_os_xxx()
  - Decoupled from platform HAL and common NS mailbox
    - Thread mgmt. is moved out from platform impl.
    - Semaphores are moved out from multi-core API

- NS mailbox RTOS APIs implementation
  - tfm_ns_mailbox_rtos_api.c as a reference
    - Can be directly replaced with RTOS specific impl.
  - RTOS API wrapper becomes optional
Easier integration with RTOS on platforms

- **Platform specific mailbox HW impl.**
  - Implemented by platform partners
  - Implemented under platform folder in TF-M

- **RTOS support to NS mailbox**
  - Implemented by application developers or platform partners according to actual usage scenarios
  - Maintain a dedicated .c for each RTOS

![Diagram showing PSA Client APIs, NS Mailbox API, NS Mailbox, NS Mailbox RTOS APIs, NS Mailbox HAL APIs, Thread mgmt., Semaphore, Platform specific mailbox impl., Implemented by RTOS or platforms, Implemented by platforms, and Independent to platform or RTOS]
Enhance NS mailbox working model
Extra port effort in a more complex usage scenario

- NS MPU enabled to isolate NS applications
  - Difficult to specify mailbox static objects addresses in application thread MPU regions
  - Modify common NS mailbox impl. to insert SVCs
  - Different SVC handler hacks in various RTOSs
- Goal: Support NS thread isolation more easily
A new NS mailbox working model

*Besides existing NS mailbox implementation*

- A dedicated NS mailbox thread assigned
  - Execute `mailbox_thread_runner()`
  - Receive requests from application threads via RTOS message queue
  - Wait if mailbox queue is full

- Simplify RTOS port for thread isolation
  - No explicit SVC is required
    - No longer necessary to hack common NS mailbox or RTOS
  - Get rid of semaphores

- Fit more in RTOS/OS thread mgmt.
  - Mailbox dedicated thread can run in privileged mode
  - Application threads can be isolated
    - Each thread maintains its own set of resource during NS mailbox handling

---

**Diagram:**

- NSPE
  - PSA Client APIs
  - `tfm_ns_mailbox_client_call()`
  - `rtos_create_mq()`
  - `rtos_mq_send()`
  - `rtos_mq_receive()`
  - `mailbox_thread_runner()`
  - Platform IPC NS IRQ Handler
    - Common handler
      - Write back result
      - Wake up
  - Triggered by SPE mailbox IPC IRQ

- SPE
  - RTOS mailbox dedicated thread
    - Prepare request
    - `wait`
    - Send Client call to SPE mailbox
Refine NS mailbox working model configuration

- Clarify the responsibilities of platform implementation and NS integration

<table>
<thead>
<tr>
<th>NUM_MAILBOX_QUEUE_SLOT &gt; 1</th>
<th>NUM_MAILBOX_QUEUE_SLOT == 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFM_MULTI_CORE_NS_OS OFF</td>
<td>NS bare metal environment</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_NS_OS ON</td>
<td>NS OS environment</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_NS_MAILBOX_THREAD OFF</td>
<td>Enable multiple PSA NS client call feature</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_NS_MAILBOX_THREAD ON</td>
<td>NS OS environment</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_OS_MAILBOX_THREAD OFF</td>
<td>NS OS environment</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_OS_MAILBOX_THREAD ON</td>
<td>Dedicated NS mailbox thread</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_MULTI_CLIENT_CALL OFF</td>
<td>Enable multiple PSA NS client call feature</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_MULTI_CLIENT_CALL ON</td>
<td>NS OS environment</td>
</tr>
<tr>
<td>NUM_MAILBOX_QUEUE_SLOT &gt; 1</td>
<td>Enable multiple PSA NS client call feature</td>
</tr>
<tr>
<td>NUM_MAILBOX_QUEUE_SLOT == 1</td>
<td>Rely on platform IPC interrupt</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_MULTI_CLIENT_CALL OFF</td>
<td>Disable multiple PSA NS client call feature</td>
</tr>
<tr>
<td>TFM_MULTI_CORE_MULTI_CLIENT_CALL ON</td>
<td>Looping mailbox flag</td>
</tr>
</tbody>
</table>
Current status
Current status

• Patches under review
  • Enhancement
  • New NS mailbox working model

• Collecting feedback from partners for actual usage scenarios
  • Comments and suggestions are welcome

• NS mailbox enhancement
  • Looking forward to achieving approval

• NS mailbox working model with a dedicated thread
  • Further discussion if necessary
Thank You
Danke
Merci
Merci
ありがとうございます
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה
Quantitative results comparison

- Latency or throughput is not affected in this proposal
  - Compared to current implementation
  - Although performance is not the main purpose in this proposal

<table>
<thead>
<tr>
<th></th>
<th>Current Impl.</th>
<th>Enhancement</th>
<th>Dedicated thread</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lightweight test</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total nr of threads</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Nr of pending slots in average</td>
<td>2.5</td>
<td>2.9</td>
<td>1.7</td>
</tr>
<tr>
<td>Ticks cost in each PSA client call</td>
<td>0.2</td>
<td>0.1</td>
<td>8.5</td>
</tr>
<tr>
<td>Ticks cost in total</td>
<td>820</td>
<td>715</td>
<td>30761</td>
</tr>
<tr>
<td><strong>Heavyweight test</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total nr of threads</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Nr of pending slots in average</td>
<td>3.8</td>
<td>3.9</td>
<td>3.8</td>
</tr>
<tr>
<td>Ticks cost in each round</td>
<td>699.3</td>
<td>697.1</td>
<td>698.1</td>
</tr>
<tr>
<td>Ticks cost in total</td>
<td>335710</td>
<td>334632</td>
<td>335109</td>
</tr>
<tr>
<td><strong>Out-of-Order test</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total nr of threads</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Nr of pending slots in average</td>
<td>2.8</td>
<td>2.8</td>
<td>2.6</td>
</tr>
<tr>
<td>Ticks cost in each round</td>
<td>11.8</td>
<td>11.0</td>
<td>12.0</td>
</tr>
<tr>
<td>Ticks cost in total</td>
<td>31450</td>
<td>29452</td>
<td>32000</td>
</tr>
</tbody>
</table>

*Based on TF-Mv1.2.0
TF-M multi-core tests running on Cypress PSoC 64
Total 4 mailbox queue slots*
Further security consideration

Not implemented yet

- “Boomerang” attack
  - SPE is unaware of corresponding NSPE isolation configs
  - NS malicious app cheats SPE to access other NS thread area or NS privileged area, bypassing NSPE MPU HW

- NS memory check in NS mailbox
  - If required by usage scenario thread model
  - Essential check: an unprivileged app provides addresses belonging to privileged areas.
  - Advanced check: an app provides addresses not belonging to itself
    - Highly depends on platform and RTOS impl.