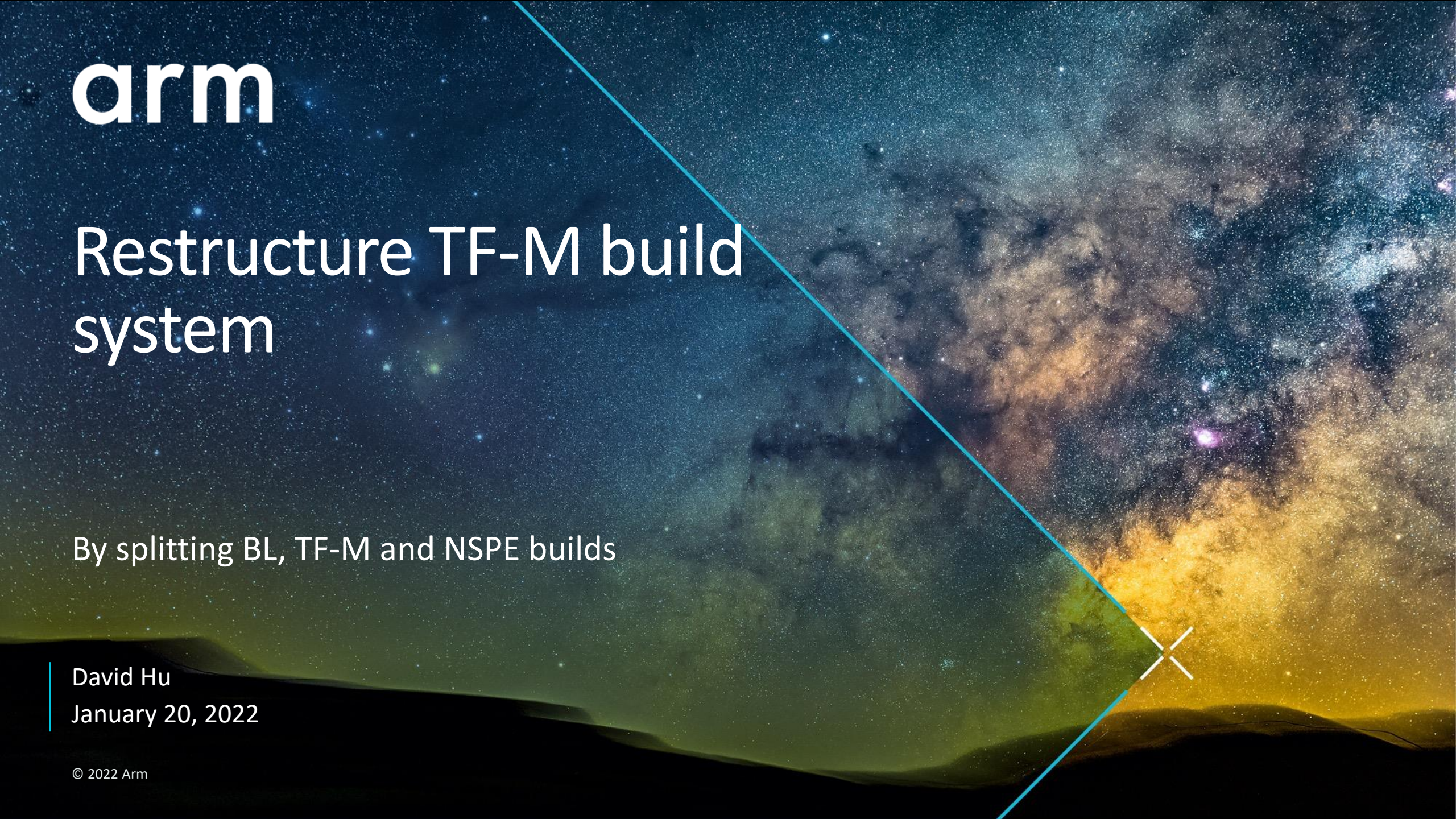


Restructure TF-M build system

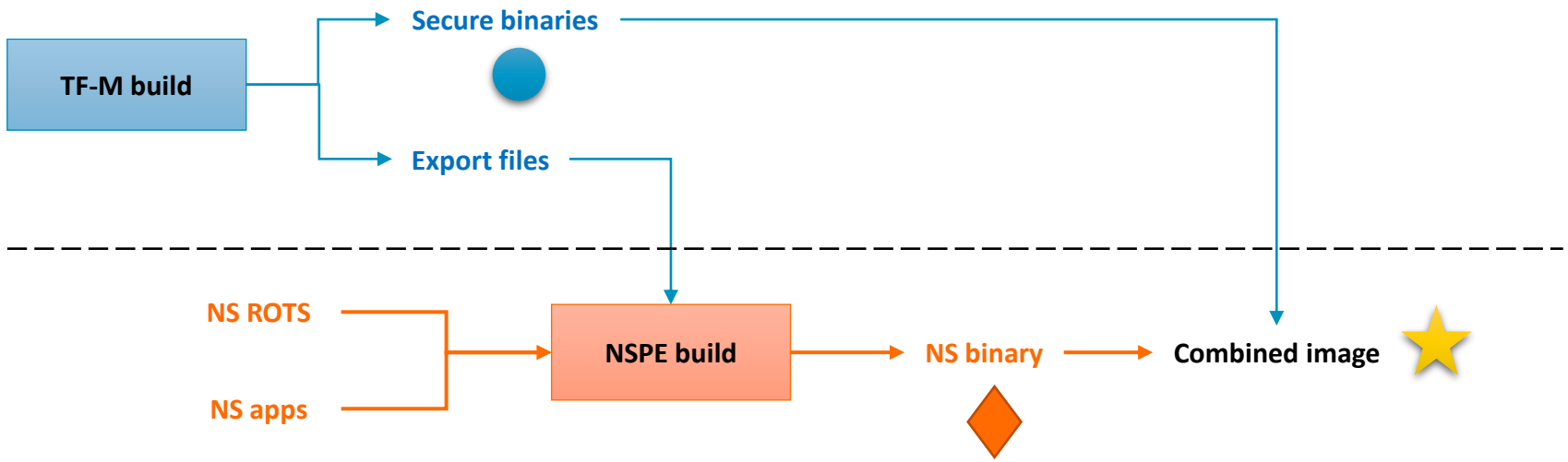
By splitting BL, TF-M and NSPE builds

David Hu

January 20, 2022

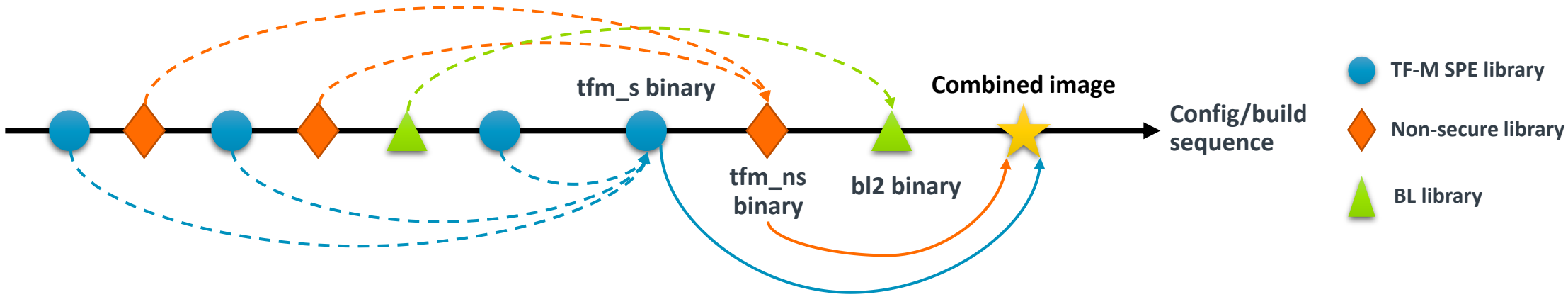
Typical build process

- Build TF-M and NSPE independently
  - A typical sequence
    - Build TF-M
    - Export files and binaries
    - Build NS RTOS and application with files exported by TF-M
    - Combine secure image and non-secure image



Limitations of current TF-M build sequence

- Integration of NS build and SPE build is simplified
  - Bootloader (BL), TF-M SPE and NSPE configs/builds are mixed
    - NS libraries are built together with secure ones
    - NS is built with TF-M source code directly, rather than with the exported/installed ones



  - *A clearer reference can help developers better understand how to integrate TF-M*

arm

Limitations of current TF-M build sequence

- Difficult to specify distinct configurations for BL, SPE and NSPE
  - Multi-core platforms: non-secure core and secure core require different configurations
    - Reload NS toolchain configuration before NS libraries are added.
    - It requires to carefully maintain CMake file structure and select the correct places to reload config

```
if (TFM_MULTI_CORE_TOPOLOGY)
    include(……/preload_ns.cmake)
    tfm_toolchain_reload_compiler()
    # The platform target is created in this directory/file
    # so that it has the same settings as the main ns target.
    add_library(platform_ns STATIC EXCLUDE_FROM_ALL)
endif()
```

  - Floating-Point feature support: dedicated FP build flags
    - Current workaround explicitly sets FP build in each TF-M library
      - Even if a library doesn't need FP support
    - Non-trivial maintenance effort

```
target_compile_options(tfm_qcbor_s
    PRIVATE
        ${COMPILER_CP_FLAG}
)
```

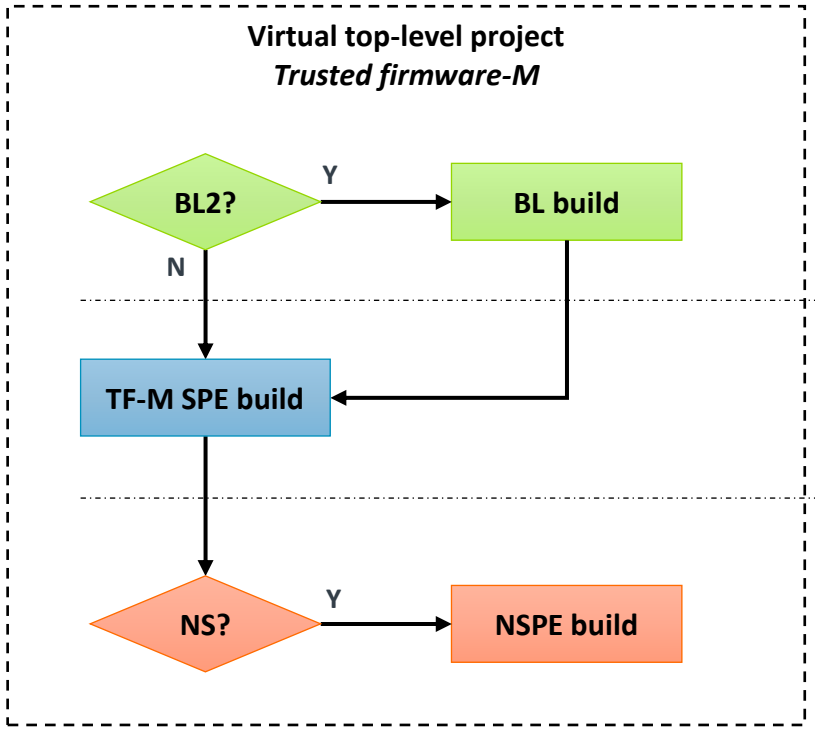
  - *Less extensible in more complex trusted system or with new security features.*

Limitations of current TF-M build sequence

- Pay attention to some issues
  - Link secure libraries to non-secure build
  - Header files not exported as request
  - Specify secure build flags for non-secure libraries, and vice versa

Proposal

- Separate builds of BL, TF-M SPE and NSPE
  - Build each module as a CMake external project under a virtual top-level project
  - Simulate actual integration scenarios




```
if(BL2)
    ExternalProject_Add(BL2
        ......
    )
endif()

ExternalProject_Add(TF-M-SPE
    ......
)

if(NS)
    ExternalProject_Add(TF-M-NSPE
        ......
    )
endif()
```

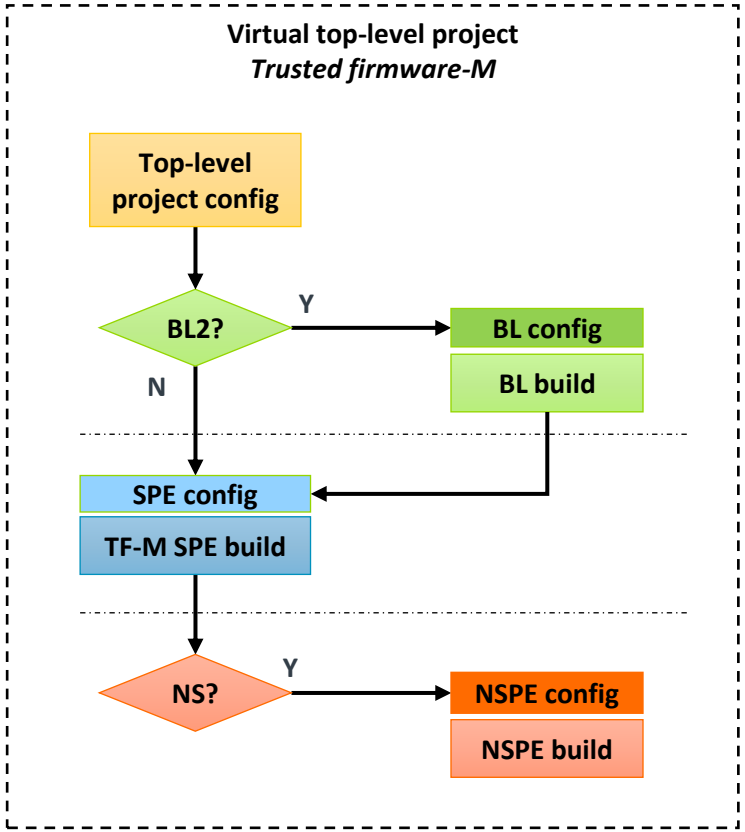
Proposal

- Goals
  - Make it easier when users integrate downstream TF-M in actual scenarios with similar build sequence
    - Less integration cost
    - Fewer surprise
  - Users can take the top-level project Cmake file as a reference
    - Replace BL2 and tf-m-tests with own bootloader and NSPE respectively

Details

Configuration process

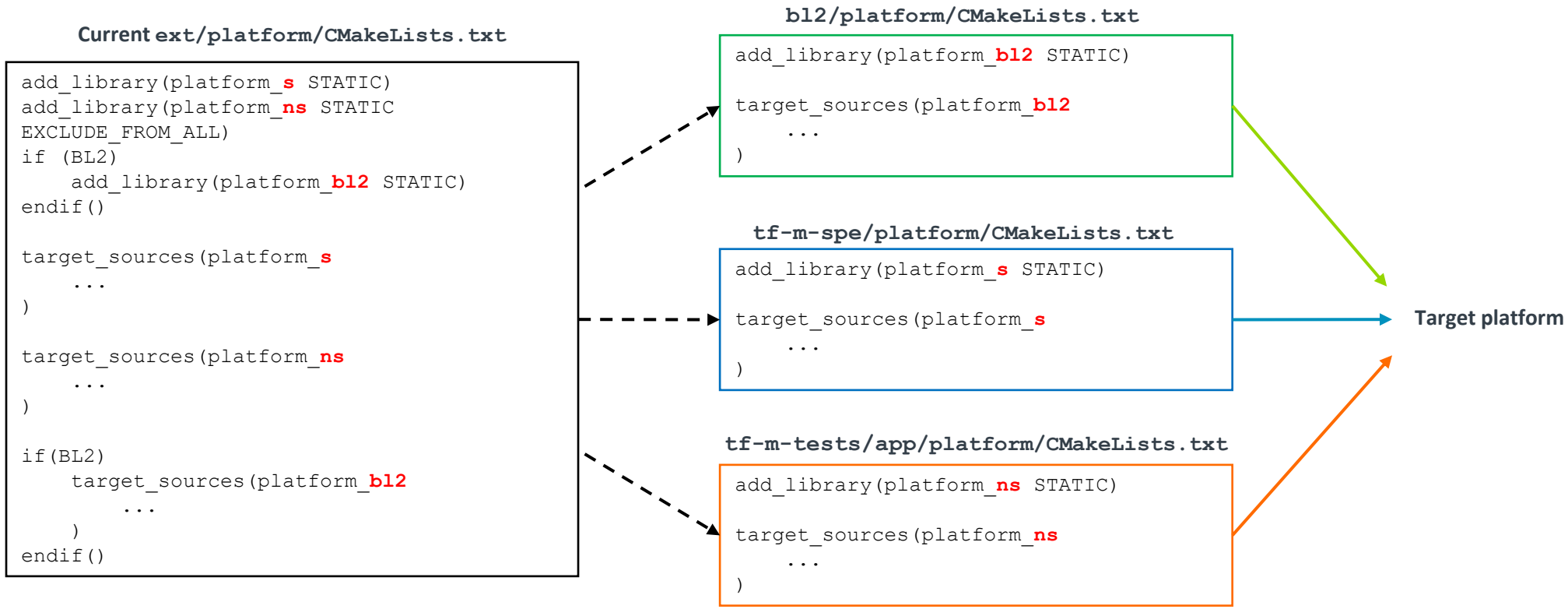
- Add a top-level project configuration step
  - Specify the configurations which impact the whole project structure
    - NS
    - BL2
    - 3rd-party open-source projects shared by multiple modules
      - tf-m-tests
      - PSA Arch test
      - Mbed TLS
- Each module performs its own configuration during its dedicated build
  - Each build can specify its dedicated configs in its own config file



Details

Changes to platforms

- Separate platform root `CMakeLists.txt`
  - BL, TF-M SPE and NSPE add dedicated platform libraries and include target respectively



Details

Changes to platform (cont'd)

- Individual platform dedicated CMake files
  - Add conditional check for building module specific libraries
  - Update file paths due to trusted-firmware-m code structure change
    - PLATFORM_DIR
    - TFM_TOP_SOURCE_DIR

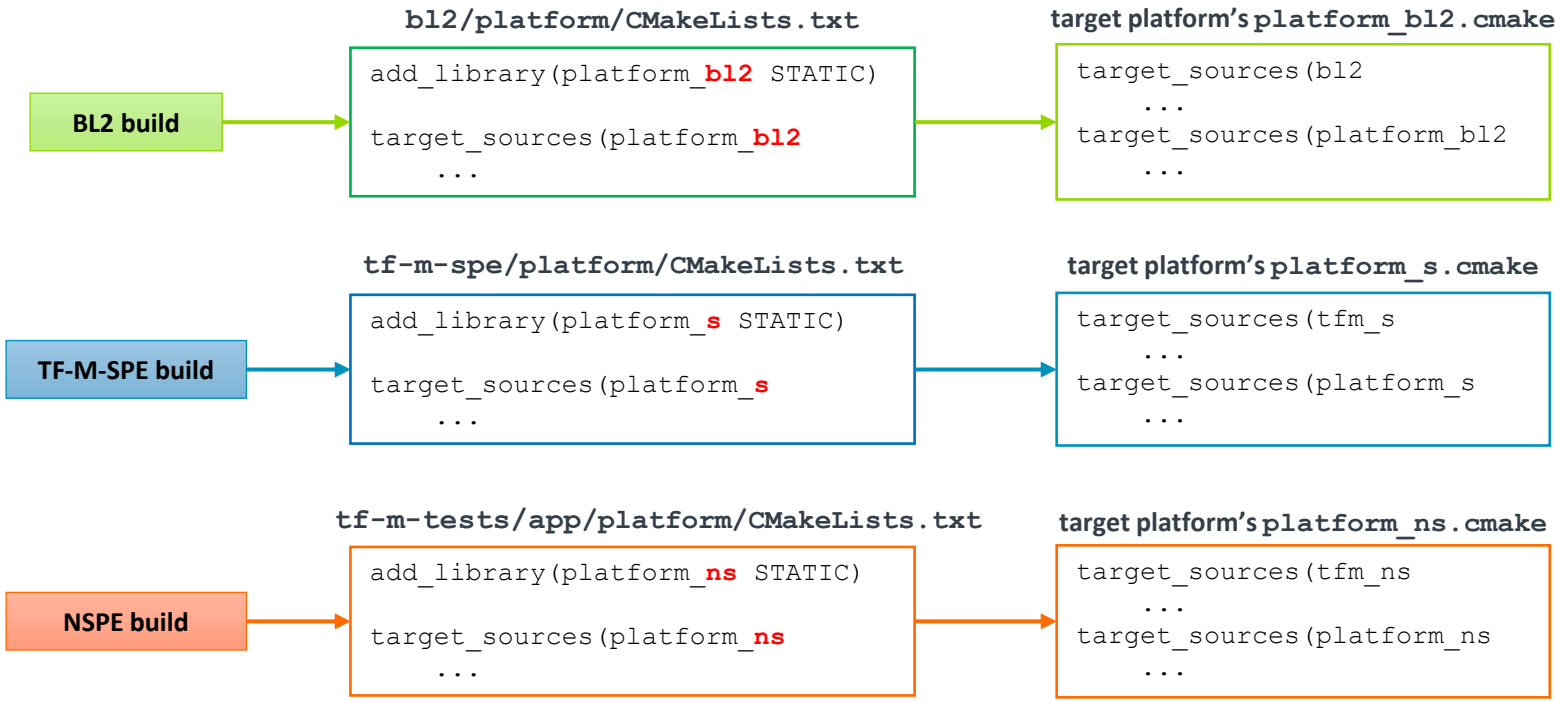
target platform's `CMakeLists.txt`

```
if(SPE_BUILD)
    target_sources(tfm_s
        ...
    )
    target_sources(platform_s
        ...
    )
    ...
endif()

if(NS_BUILD)
    target_sources(tfm_ns
        ...
    )
    target_sources(platform_ns
        ...
    )
    ...
endif()

if(BL_BUILD)
    target_sources(bl2
        ...
    )
    target_sources(platform_bl2
        ...
    )
    ...
endif()
```

Details

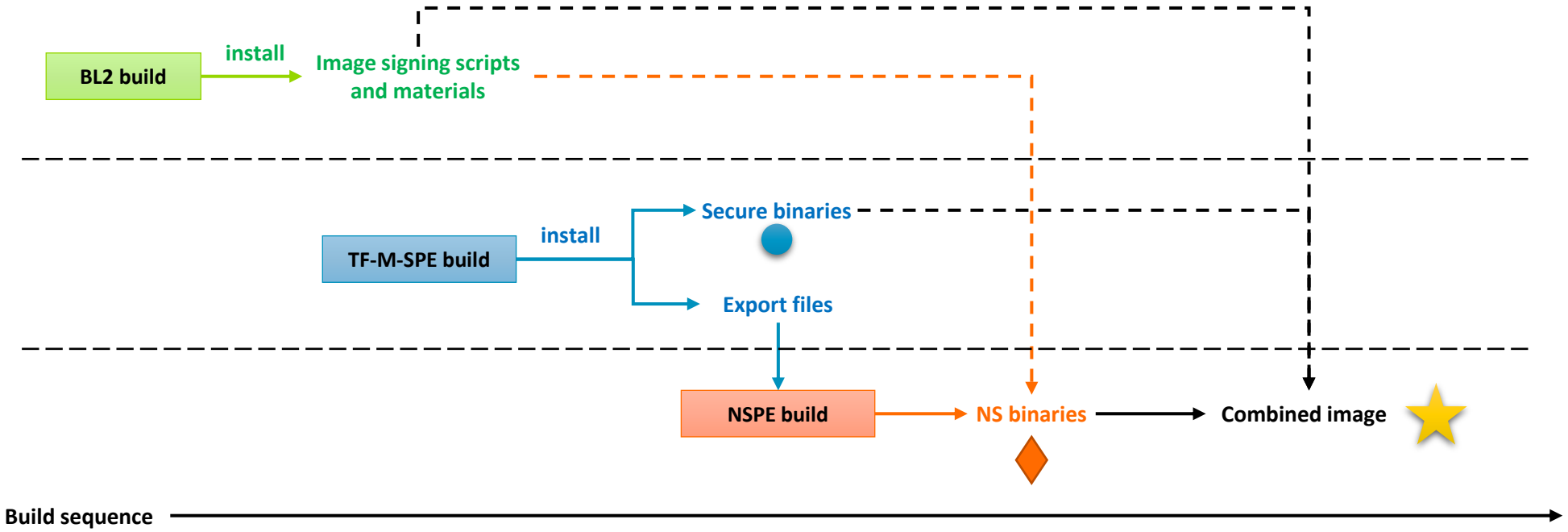
Changes to platform (cont'd)

- Individual platform dedicated CMake files
  - Alternative: split SPE/NSPE/BL2 builds as well
    - More clean but more complex
    - Update file paths as well



Details

- Install and image generation process
  - NSPE is built with files exported by TF-M SPE build
  - Secure image and non-secure image are generated during SPE build and NSPE build respectively
  - NS and S images are combined by NSPE build if BL2 is enabled



Details

- Users won’t be aware of changes while building TF-M
  - Build commands are kept the same
  - Configurations are unchanged
  - *Except* build output logs
    - (Imoo) easier for debugging
      - Logs are not mixed anymore
      - Builds terminate immediately after the fatal error occurs
    - However, configuration messages might be duplicated
      - Shall be sorted and simplified further

Current status

- A PoC under review
  - Patch set
    - trusted-firmware-m patch set
    - tf-m-tests patch set
  - Most major features are tested. All platforms are built successfully.
  - It will be rebased (reworked) after other restructure patches are merged. Implementation details might be changed then.
  - Comments are welcome!

Further improvements

- More flexible configuration settings passed among modules
- Decouple image signing and assemble from TF-M SPE and NSPE build
- Perhaps further separation of TF-M repos

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