Proofcraft

The Next 700 Verified seL4 Platforms

Part of the INSPECTA project in the DARPA PROVERS program

Gerwin Klein | seL4 summit 2025, Prague, Czech Republic

4

Who needs 700 platforms?

Platforms and configurations

seL4 runs on:

- Arm:
  - Avnet MaaXBoard
  - BeagleBoard
  - BeagleBone Black
  - IMX8MM-EVK
  - Odroid-C2
  - Odroid-C4
  - OdroidX
  - OdroidXU
  - OdroidXU4
  - Raspberry Pi 3b
  - Raspberry Pi 4B
  - Rockpro64
  - Sabre Lite
  - TK1
  - TK1-SOM
  - TX1
  - TX2
  - Ultra96v2
  - Zynq ZCU102
  - Zynq-7000
  - IMX8MQ
  - ZCU106
- RISC-V:
  - Ariane
  - Cheshire
  - HiFive Unleashed
  - Microchip PolarFire
  - Rocketchip
- Intel:
  - 32 bit PC-99
  - 64 bit PC-99

Verified seL4 platforms: last year

- Arm:
  - Avnet MaaXBoard
  - BeagleBoard
  - BeagleBone Black
  - IMX8MM-EVK
  - Odroid-C2
  - Odroid-C4
  - OdroidX
  - OdroidXU
  - OdroidXU4
  - Raspberry Pi 3b
  - Raspberry Pi 4B
  - Rockpro64
  - ✓ Sabre Lite
  - ✓ TK1
  - TK1-SOM
  - TX1
  - ✓ TX2
  - Ultra96v2
  - Zynq ZCU102
  - Zynq-7000
  - IMX8MQ
  - ZCU106
- RISC-V:
  - Ariane
  - Cheshire
  - ✓ HiFive Unleashed
  - Microchip PolarFire
  - Rocketchip
- Intel:
  - 32 bit PC-99
  - ✓ 64 bit PC-99

Verified seL4 platforms: now

- Arm:
  - ✓Avnet MaaXBoard
  - ✓BeagleBoard
  - ✓BeagleBone Black
  - ✓IMX8MM-EVK
  - ✓Odroid-C2
  - ✓Odroid-C4
  - ✓OdroidX
  - ✓OdroidXU
  - ✓OdroidXU4
  - ✓Raspberry Pi 3b
  - ✓Raspberry Pi 4B
  - ✓Rockpro64
  - ✓Sabre Lite
  - ✓TK1
  - ✓TK1-SOM
  - ✓TX1
  - ✓TX2
  - ✓Ultra96v2
  - ✓Zynq ZCU102
  - ✓Zynq-7000
  - ✓IMX8MQ
  - ✓ZCU106

New: ✓IMX93

- RISC-V:
  - Ariane
  - Cheshire
  - ✓HiFive Unleashed
  - Microchip PolarFire
  - Rocketchip

- Intel:
  - 32 bit PC-99
  - ✓64 bit PC-99

Verified seL4 platforms: now

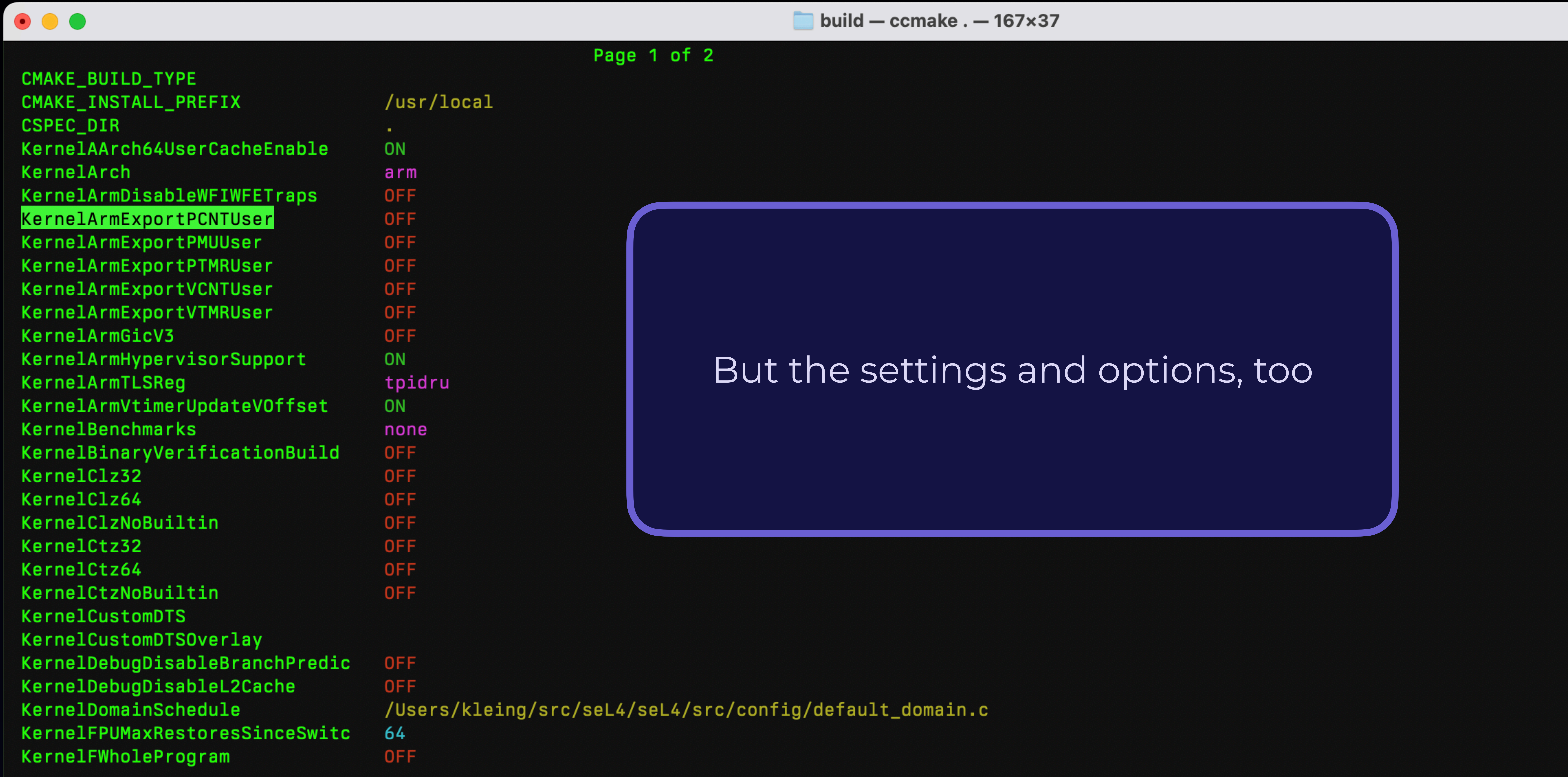
- Arm:
  - ✓ Avnet MaaXBoard
  - ✓ BeagleBoard
  - ✓ BeagleBone Black
  - ✓ IMX8MM-EVK
  - ✓ Odroid-C2
  - ✓ Odroid-C4
  - ✓ OdroidX
  - ✓ OdroidXU
  - ✓ OdroidXU4
  - ✓ Raspberry Pi 3b
  - ✓ Raspberry Pi 4B
  - ✓ Rockpro64
  - ✓ Sabre Lite
  - ✓ TK1
  - ✓ TK1-SOM
  - ✓ TX1
  - ✓ TX2
  - ✓ Ultra96v2
  - ✓ Zynq ZCU102
  - ✓ Zynq-7000
  - ✓ IMX8MQ
  - ✓ ZCU106

New: ✓ IMX93

All Arm platforms now have verification support.

- Intel:
  - 32 bit PC-99
  - ✓ 64 bit PC-99

Future: not just the platforms



build — ccmake . — 167×37

```
                                                  Page 1 of 2
 CMAKE_BUILD_TYPE
 CMAKE_INSTALL_PREFIX             /usr/local
 CSPEC_DIR                        .
 KernelAArch64UserCacheEnable     ON
 KernelArch                       arm
 KernelArmDisableWFIWFETraps      OFF
 KernelArmExportPCNTUser          OFF
 KernelArmExportPMUUser           OFF
 KernelArmExportPTMRUser          OFF
 KernelArmExportVCNTUser          OFF
 KernelArmExportVTMRUser          OFF
 KernelArmGicV3                   OFF
 KernelArmHypervisorSupport       ON
 KernelArmTLSReg                  tpidru
 KernelArmVtimerUpdateVOffset     ON
 KernelBenchmarks                 none
 KernelBinaryVerificationBuild    OFF
 KernelClz32                      OFF
 KernelClz64                      OFF
 KernelClzNoBuiltin               OFF
 KernelCtz32                      OFF
 KernelCtz64                      OFF
 KernelCtzNoBuiltin               OFF
 KernelCustomDTS
 KernelCustomDTSOverlay
 KernelDebugDisableBranchPredic   OFF
 KernelDebugDisableL2Cache        OFF
 KernelDomainSchedule             /Users/kleing/src/seL4/seL4/src/config/default_domain.c
 KernelFPUMaxRestoresSinceSwitc   64
 KernelFWholeProgram              OFF
```

But the settings and options, too

Not really 700 kernels

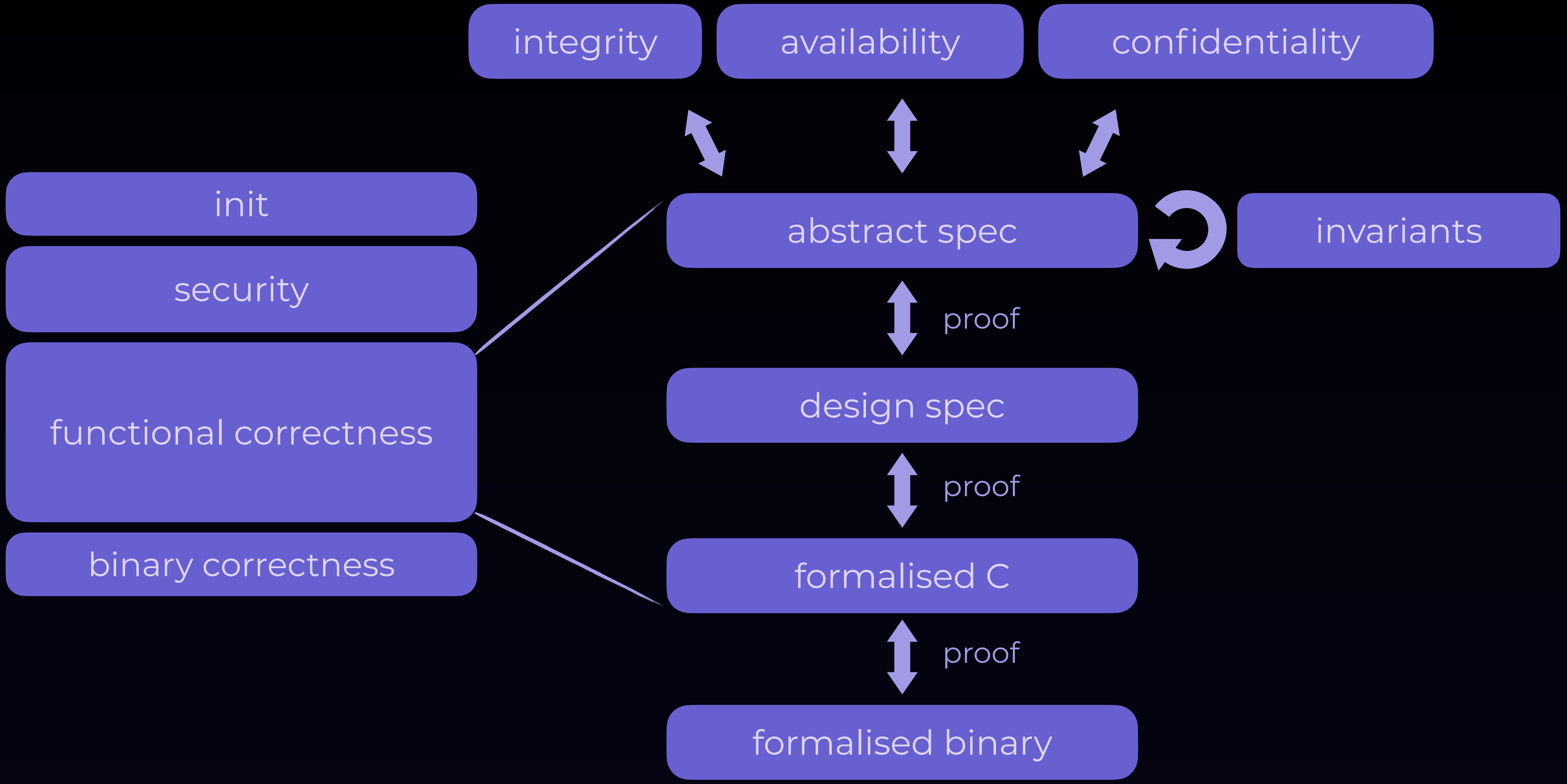
Closer to 1,099,511,627,776 kernels

4

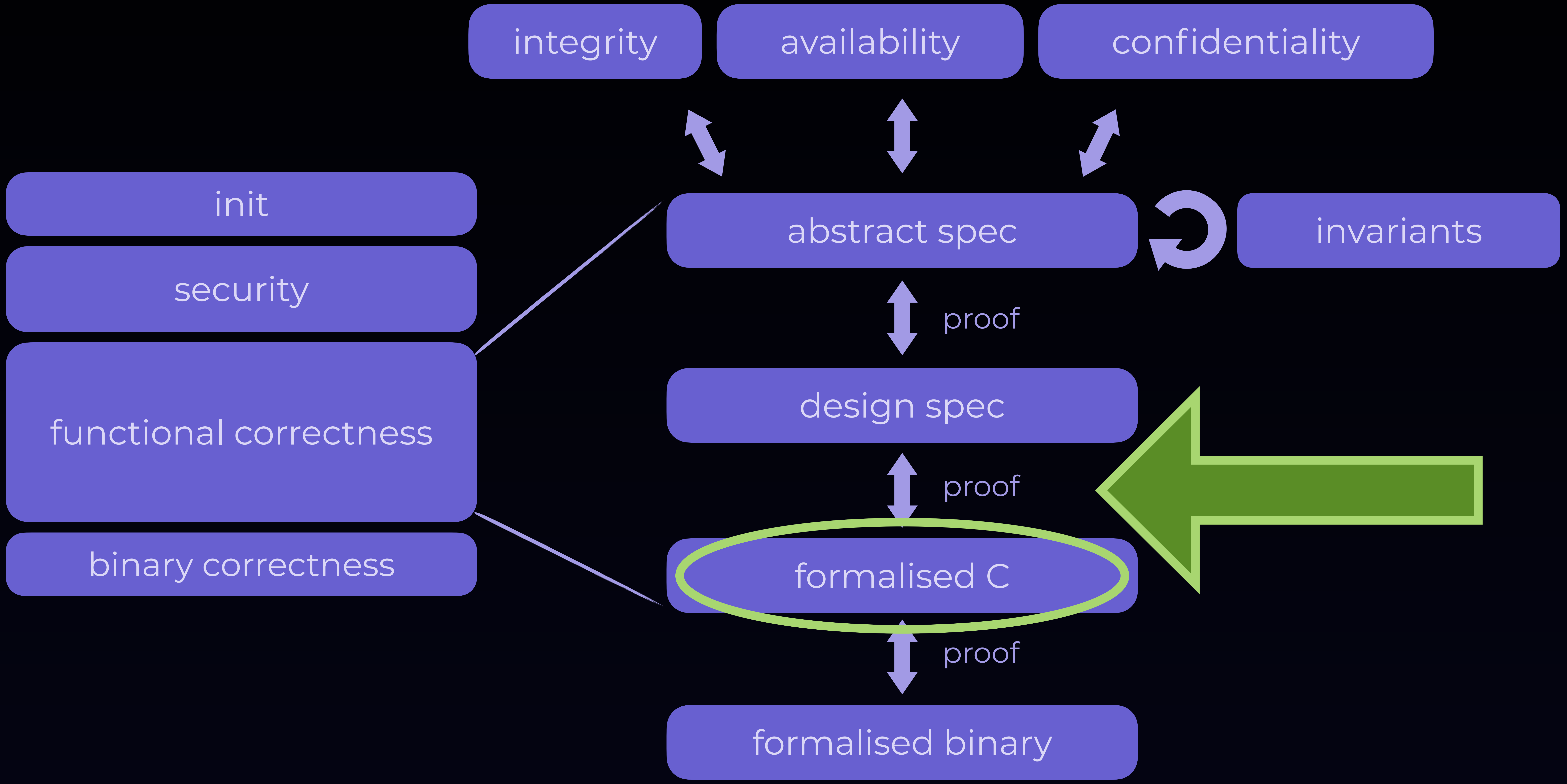
What’s behind this?

Verifying with Conditional Compilation

Verification stack

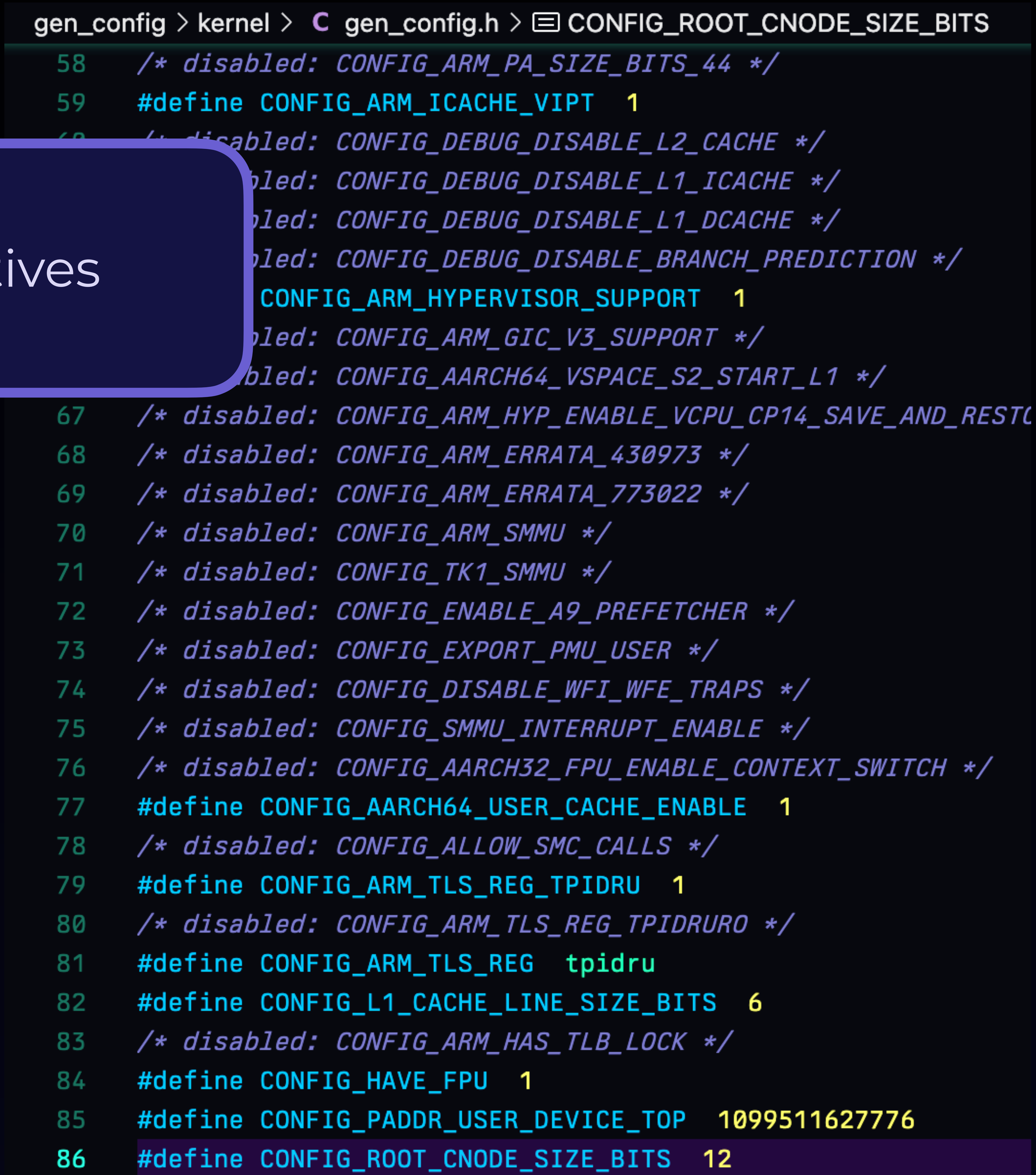


Verification stack



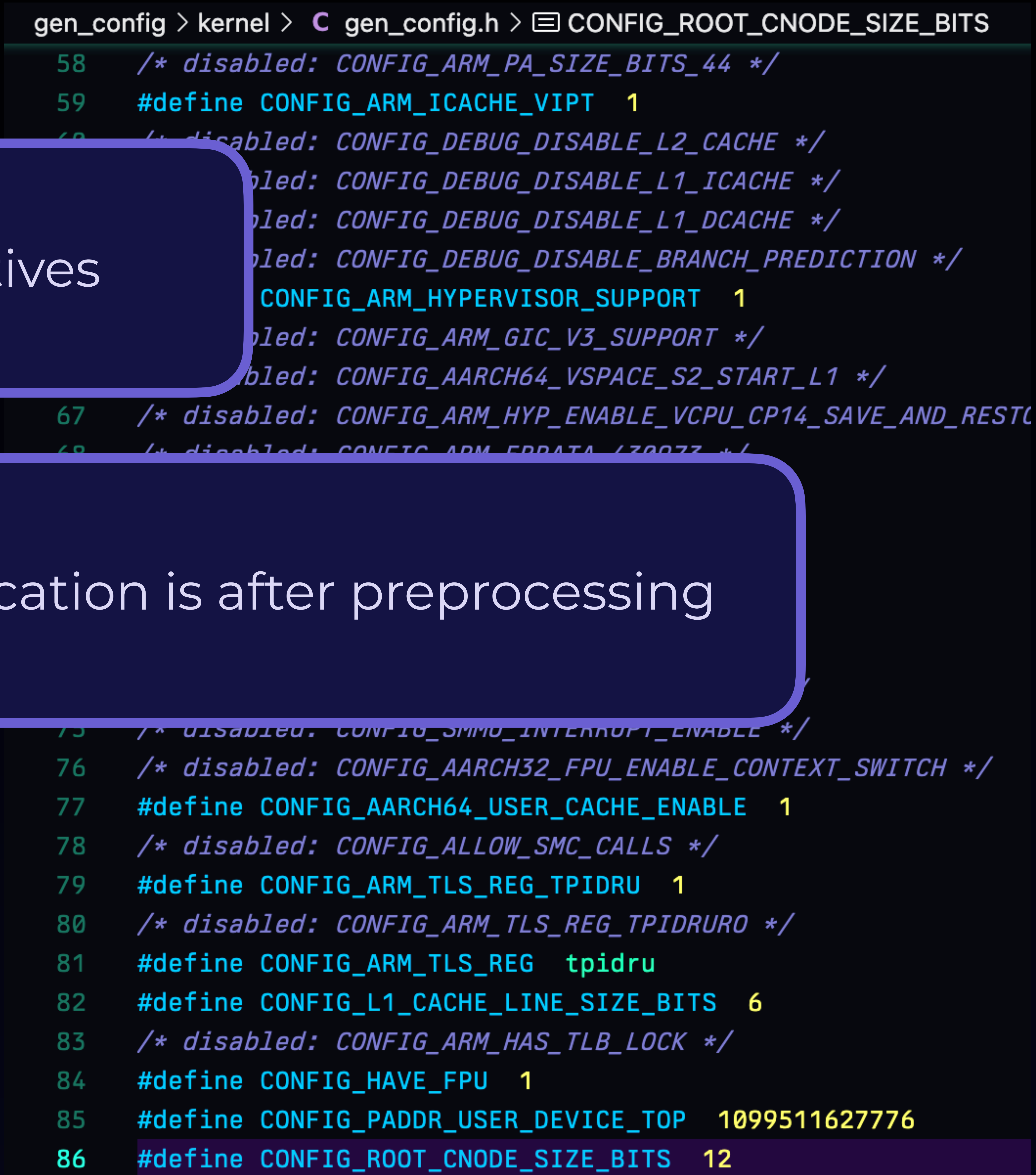
Configs use conditional compilation

Options become preprocessor directives


gen_config > kernel > C gen_config.h > CONFIG_ROOT_CNODE_SIZE_BITS

```
/* disabled: CONFIG_ARM_PA_SIZE_BITS_44 */
#define CONFIG_ARM_ICACHE_VIPT  1
/* disabled: CONFIG_DEBUG_DISABLE_L2_CACHE */
bled: CONFIG_DEBUG_DISABLE_L1_ICACHE */
bled: CONFIG_DEBUG_DISABLE_L1_DCACHE */
bled: CONFIG_DEBUG_DISABLE_BRANCH_PREDICTION */
 CONFIG_ARM_HYPERVISOR_SUPPORT  1
bled: CONFIG_ARM_GIC_V3_SUPPORT */
bled: CONFIG_AARCH64_VSPACE_S2_START_L1 */
/* disabled: CONFIG_ARM_HYP_ENABLE_VCPU_CP14_SAVE_AND_REST
/* disabled: CONFIG_ARM_ERRATA_430973 */
/* disabled: CONFIG_ARM_ERRATA_773022 */
/* disabled: CONFIG_ARM_SMMU */
/* disabled: CONFIG_TK1_SMMU */
/* disabled: CONFIG_ENABLE_A9_PREFETCHER */
/* disabled: CONFIG_EXPORT_PMU_USER */
/* disabled: CONFIG_DISABLE_WFI_WFE_TRAPS */
/* disabled: CONFIG_SMMU_INTERRUPT_ENABLE */
/* disabled: CONFIG_AARCH32_FPU_ENABLE_CONTEXT_SWITCH */
#define CONFIG_AARCH64_USER_CACHE_ENABLE  1
/* disabled: CONFIG_ALLOW_SMC_CALLS */
#define CONFIG_ARM_TLS_REG_TPIDRU  1
/* disabled: CONFIG_ARM_TLS_REG_TPIDRURO */
#define CONFIG_ARM_TLS_REG  tpidru
#define CONFIG_L1_CACHE_LINE_SIZE_BITS  6
/* disabled: CONFIG_ARM_HAS_TLB_LOCK */
#define CONFIG_HAVE_FPU  1
#define CONFIG_PADDR_USER_DEVICE_TOP  1099511627776
#define CONFIG_ROOT_CNODE_SIZE_BITS  12
```

Configs use conditional compilation



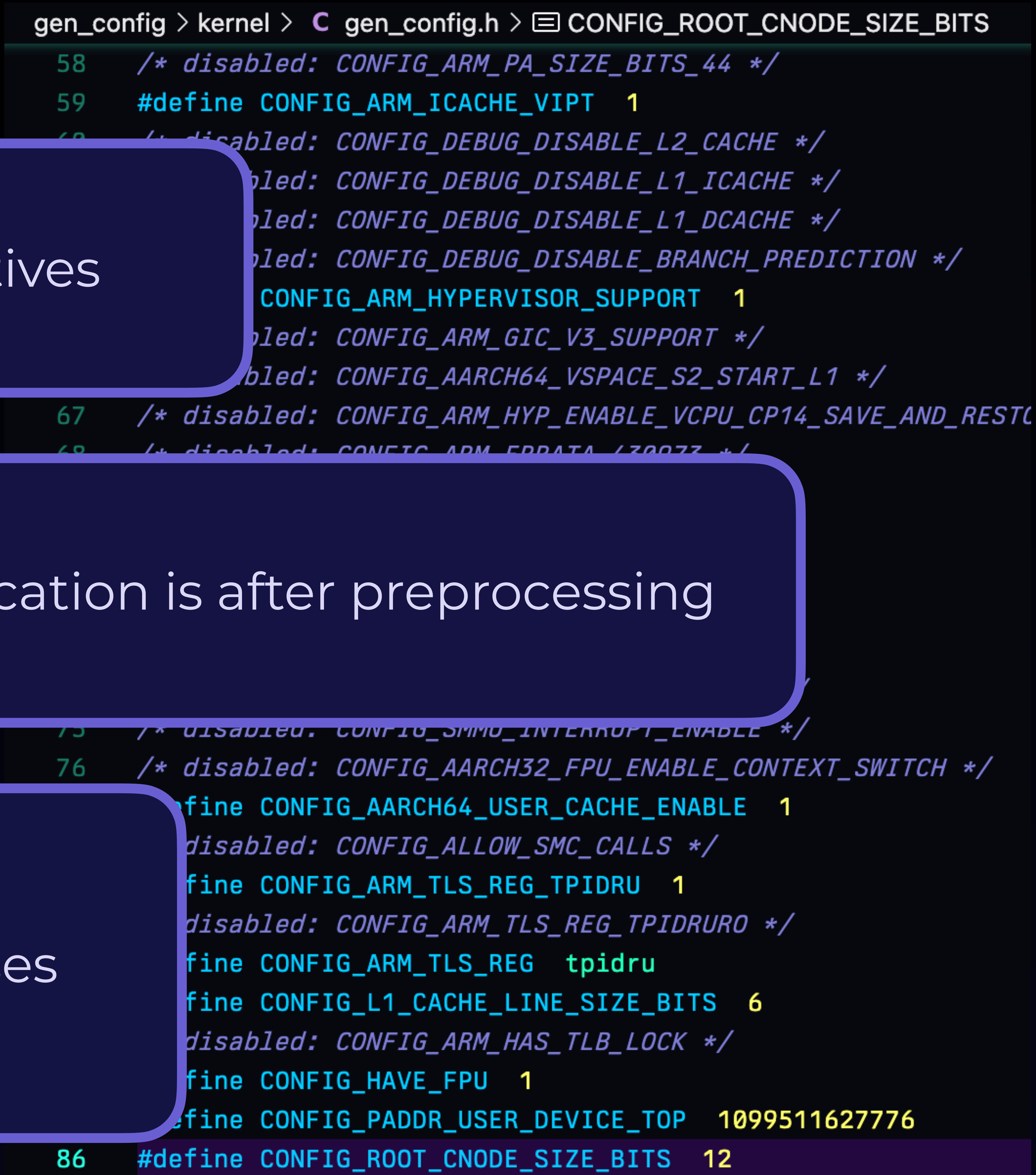
gen_config > kernel > C gen_config.h > CONFIG_ROOT_CNODE_SIZE_BITS

```
/* disabled: CONFIG_ARM_PA_SIZE_BITS_44 */
#define CONFIG_ARM_ICACHE_VIPT  1
/* disabled: CONFIG_DEBUG_DISABLE_L2_CACHE */
bled: CONFIG_DEBUG_DISABLE_L1_ICACHE */
bled: CONFIG_DEBUG_DISABLE_L1_DCACHE */
bled: CONFIG_DEBUG_DISABLE_BRANCH_PREDICTION */
CONFIG_ARM_HYPERVISOR_SUPPORT  1
bled: CONFIG_ARM_GIC_V3_SUPPORT */
bled: CONFIG_AARCH64_VSPACE_S2_START_L1 */
/* disabled: CONFIG_ARM_HYP_ENABLE_VCPU_CP14_SAVE_AND_RESTC
/* disabled: CONFIG_ARM_ERRATA_430973 */

/* disabled: CONFIG_SMMU_INTERRUPT_ENABLE */
/* disabled: CONFIG_AARCH32_FPU_ENABLE_CONTEXT_SWITCH */
#define CONFIG_AARCH64_USER_CACHE_ENABLE  1
/* disabled: CONFIG_ALLOW_SMC_CALLS */
#define CONFIG_ARM_TLS_REG_TPIDRU  1
/* disabled: CONFIG_ARM_TLS_REG_TPIDRURO */
#define CONFIG_ARM_TLS_REG  tpidru
#define CONFIG_L1_CACHE_LINE_SIZE_BITS  6
/* disabled: CONFIG_ARM_HAS_TLB_LOCK */
#define CONFIG_HAVE_FPU  1
#define CONFIG_PADDR_USER_DEVICE_TOP  1099511627776
#define CONFIG_ROOT_CNODE_SIZE_BITS  12
```

Options become preprocessor directives

C verification is after preprocessing

Configs use conditional compilation



gen_config > kernel > C gen_config.h > CONFIG_ROOT_CNODE_SIZE_BITS

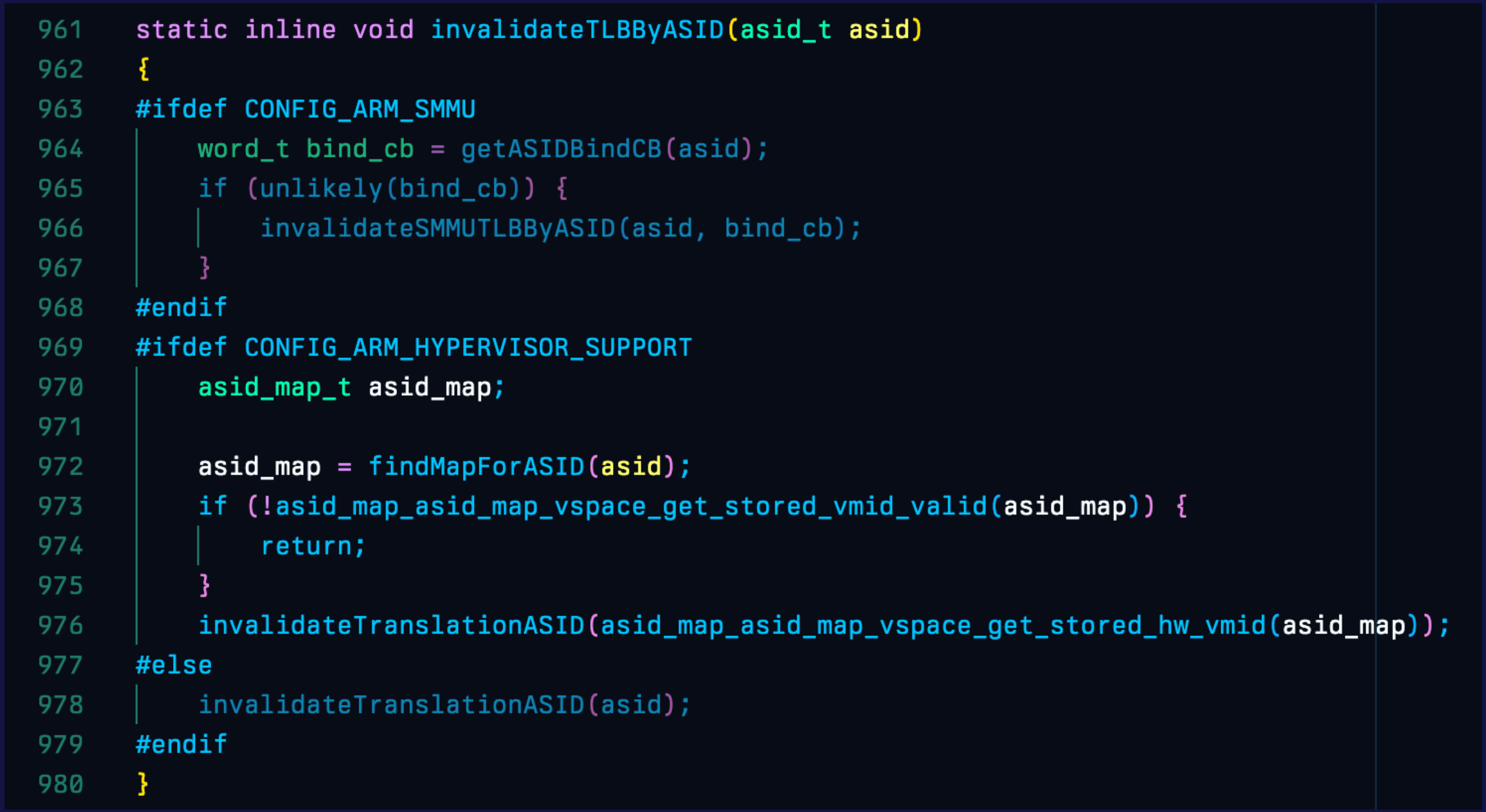
```
/* disabled: CONFIG_ARM_PA_SIZE_BITS_44 */
#define CONFIG_ARM_ICACHE_VIPT  1
/* disabled: CONFIG_DEBUG_DISABLE_L2_CACHE */
bled: CONFIG_DEBUG_DISABLE_L1_ICACHE */
bled: CONFIG_DEBUG_DISABLE_L1_DCACHE */
bled: CONFIG_DEBUG_DISABLE_BRANCH_PREDICTION */
 CONFIG_ARM_HYPERVISOR_SUPPORT  1
bled: CONFIG_ARM_GIC_V3_SUPPORT */
bled: CONFIG_AARCH64_VSPACE_S2_START_L1 */
/* disabled: CONFIG_ARM_HYP_ENABLE_VCPU_CP14_SAVE_AND_RESTO
/* disabled: CONFIG_ARM_ERRATA_430973 */

/* disabled: CONFIG_SMMU_INTERRUPT_ENABLE */
/* disabled: CONFIG_AARCH32_FPU_ENABLE_CONTEXT_SWITCH */
efine CONFIG_AARCH64_USER_CACHE_ENABLE  1
disabled: CONFIG_ALLOW_SMC_CALLS */
fine CONFIG_ARM_TLS_REG_TPIDRU  1
disabled: CONFIG_ARM_TLS_REG_TPIDRURO */
fine CONFIG_ARM_TLS_REG  tpidru
fine CONFIG_L1_CACHE_LINE_SIZE_BITS  6
disabled: CONFIG_ARM_HAS_TLB_LOCK */
fine CONFIG_HAVE_FPU  1
efine CONFIG_PADDR_USER_DEVICE_TOP  1099511627776
#define CONFIG_ROOT_CNODE_SIZE_BITS  12
```

Options become preprocessor directives

C verification is after preprocessing

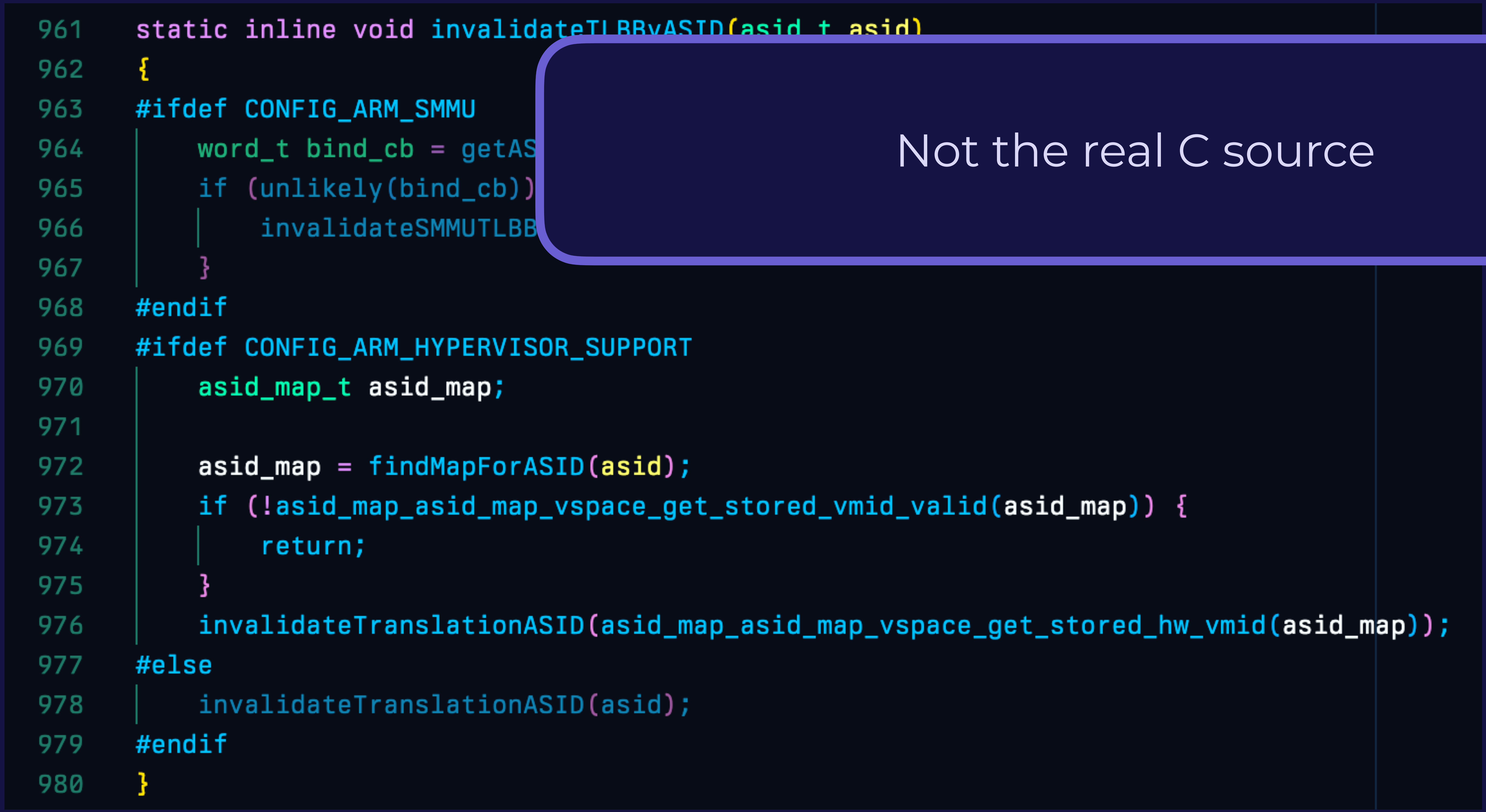
Prover sees different code bases

Conditional compilation



```
static inline void invalidateTLBByASID(asid_t asid)
{
#ifdef CONFIG_ARM_SMMU
    word_t bind_cb = getASIDBindCB(asid);
    if (unlikely(bind_cb)) {
        invalidateSMMUTLBByASID(asid, bind_cb);
    }
#endif
#ifdef CONFIG_ARM_HYPERVISOR_SUPPORT
    asid_map_t asid_map;

    asid_map = findMapForASID(asid);
    if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
        return;
    }
    invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
#else
    invalidateTranslationASID(asid);
#endif
}
```

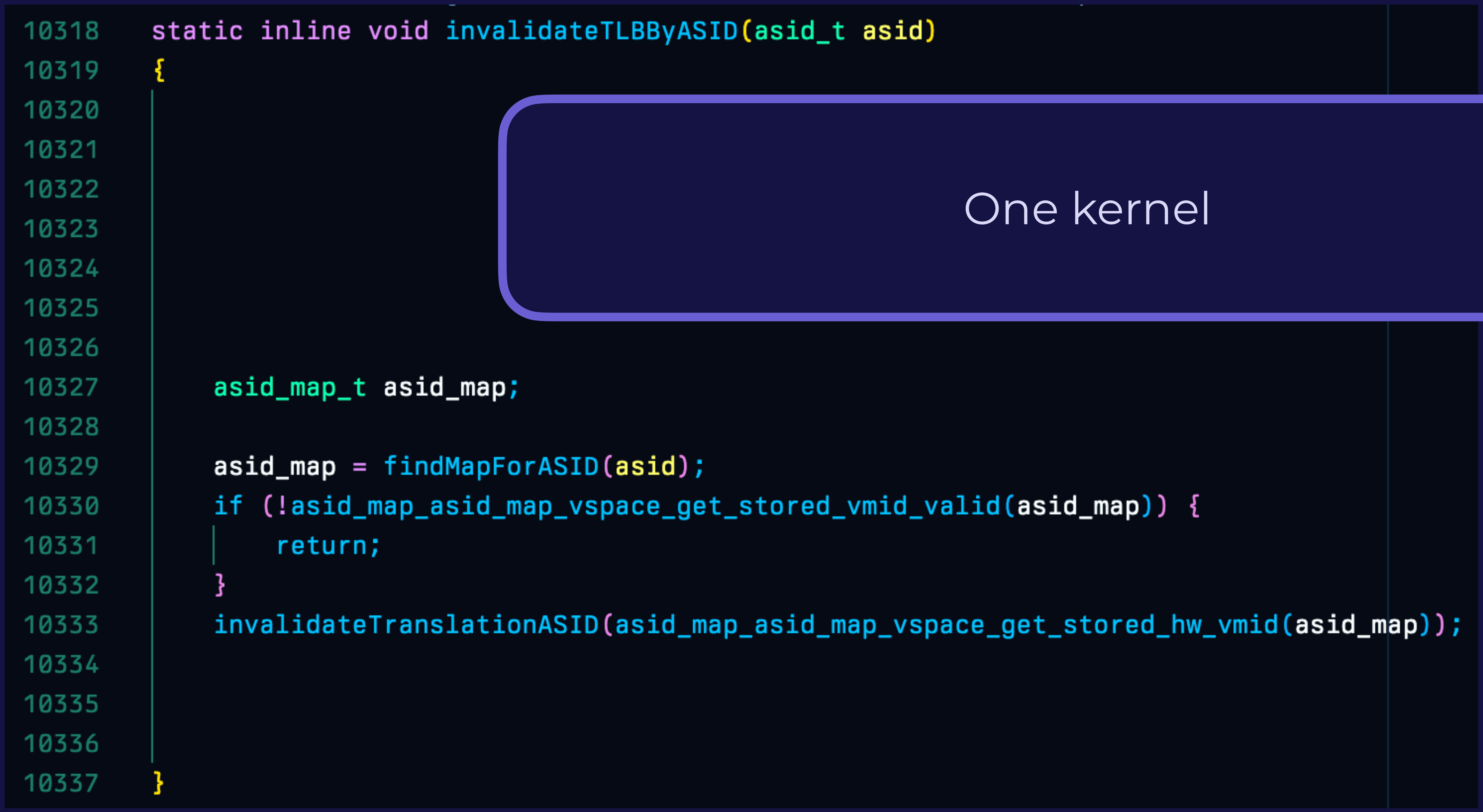
Conditional compilation



```
static inline void invalidateTLBByASID(asid_t asid)
{
#ifdef CONFIG_ARM_SMMU
    word_t bind_cb = getAS
    if (unlikely(bind_cb))
        invalidateSMMUTLBB
    }
#endif
#ifdef CONFIG_ARM_HYPERVISOR_SUPPORT
    asid_map_t asid_map;

    asid_map = findMapForASID(asid);
    if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
        return;
    }
    invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
#else
    invalidateTranslationASID(asid);
#endif
}
```

Not the real C source

Conditional compilation

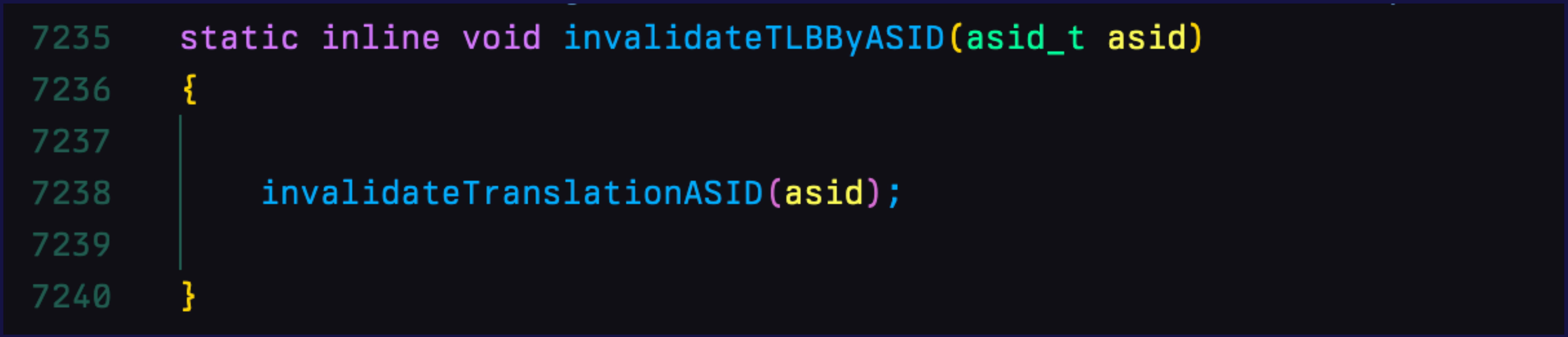


```
static inline void invalidateTLBByASID(asid_t asid)
{








    asid_map_t asid_map;

    asid_map = findMapForASID(asid);
    if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
        return;
    }
    invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));



}
```

One kernel

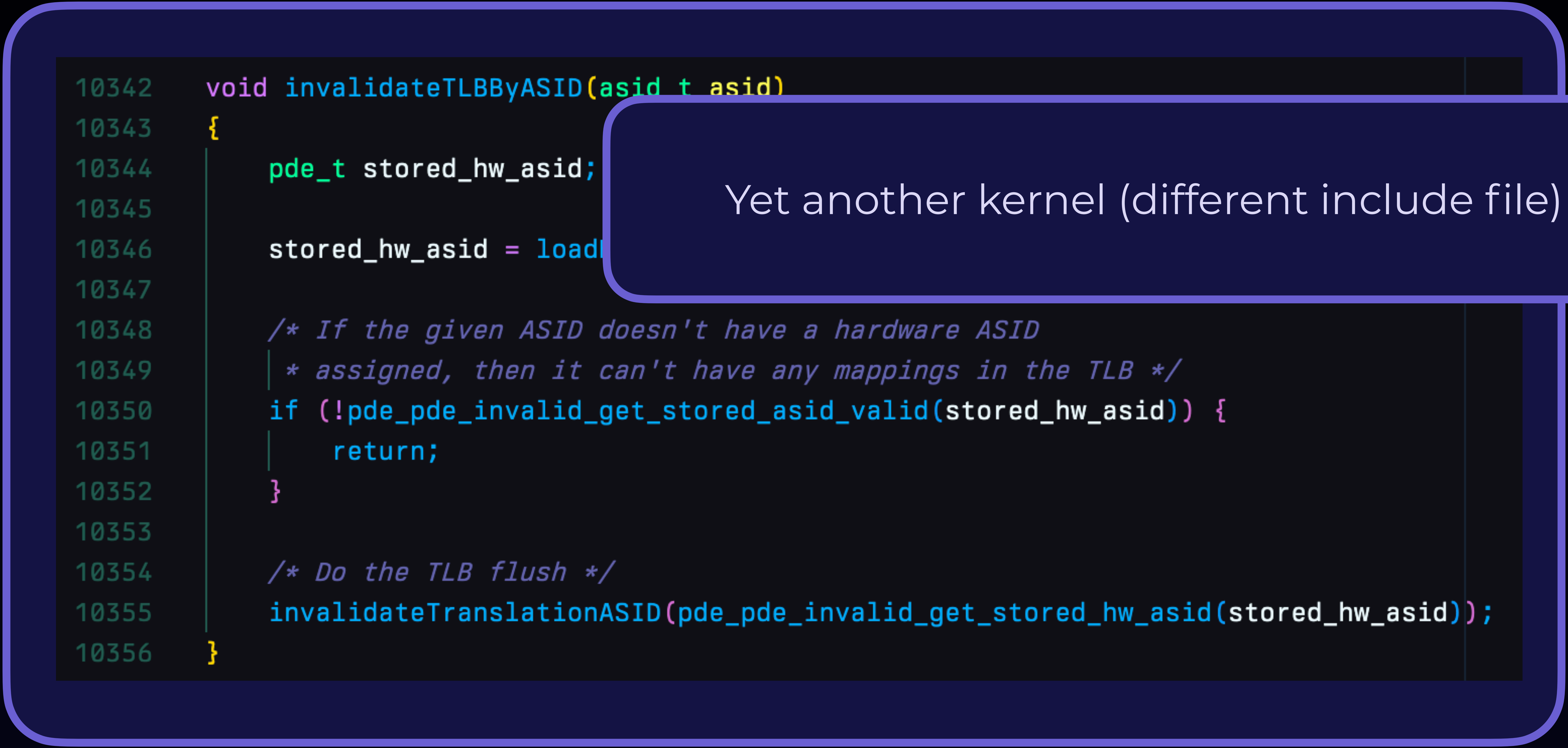
Conditional compilation

Another kernel



```
static inline void invalidateTLBByASID(asid_t asid)
{

    invalidateTranslationASID(asid);

}
```

Conditional compilation



```
void invalidateTLBByASID(asid_t asid)
{
    pde_t stored_hw_asid;

    stored_hw_asid = load

    /* If the given ASID doesn't have a hardware ASID
     * assigned, then it can't have any mappings in the TLB */
    if (!pde_pde_invalid_get_stored_asid_valid(stored_hw_asid)) {
        return;
    }

    /* Do the TLB flush */
    invalidateTranslationASID(pde_pde_invalid_get_stored_hw_asid(stored_hw_asid));
}
```

Yet another kernel (different include file)

C verification is after preprocessing

Why?

C preprocessor =
text replacement engine

C verification is after preprocessing

Why?

C preprocessor =
text replacement engine

Simple uses

```
#define SOME_CONFIG 1
#define SOME_VAL 1024
#define MAX(a,b) (((a)>(b))?(a):(b))
```

Token operations, not AST operations

```
_is_set(SOME_CONFIG) ~> 1
_is_set(OTHER_CONFIG) ~> 0
```

(instead of not evaluating)

Token operations, not AST operations

```
#define _is_set_(value) _is_set__(_macrotest_##value)
#define _is_set__(comma) _is_set___(comma 1, 0)
#define _is_set___(_, v, ...) v
#define _macrotest_1 ,
```

```
_is_set(SOME_CONFIG) ~> 1
_is_set(OTHER_CONFIG) ~> 0
```

(instead of not evaluating)

Token operations, not AST operations

```
#define _is_set_(value) _is_set__(_macrotest_##value)
#define _is_set__(comma) _is_set___(comma 1, 0)
#define _is_set___(_, v, ...) v
#define _macrotest_1 ,
```

Concatenation

```
_is_set(SOME_CONFIG) ~> 1
_is_set(OTHER_CONFIG) ~> 0
```

(instead of not evaluating)

Token operations, not AST operations

```
#define _is_set_(value) _is_set__(_macrotest_##value)
#define _is_set__(comma) _is_set___(comma 1, 0)
#define _is_set___(_, v, ...) v
#define _macrotest_1 ,
```

Concatenation

0 or more further arguments

```
_is_set(SOME_CONFIG) ~> 1
_is_set(OTHER_CONFIG) ~> 0
```

(instead of not evaluating)

Token operations, not AST operations

```
#define _is_set_(value) _is_set__(_macrotest_##value)
#define _is_set__(comma) _is_set___(comma 1, 0)
#define _is_set___(_, v, ...) v
#define _macrotest_1 ,
```

```
_is_set_(SOME_CONFIG)   ->
_is_set__(_macrotest_1) ->
_is_set__(,)            ->
_is_set___(, 1, 0)      ->

1
```

Token operations, not AST operations

```
#define _is_set_(value) _is_set__(_macrotest_##value)
#define _is_set__(comma) _is_set___(comma 1, 0)
#define _is_set___(_, v, ...) v
#define _macrotest_1 ,
```

```
_is_set_(SOME_CONFIG)   ->
_is_set__(_macrotest_1) ->
_is_set__(,)            ->
_is_set___(, 1, 0)      ->

1
```

```
_is_set_(OTHER)                       ->
_is_set__(_macrotest_OTHER)           ->
_is_set___(_macrotest_OTHER 1, 0) ->

0
```

No need to be syntactically correct

```
if (SMP_COND_STATEMENT(sc->scCore != getCurrentCPUIndex() ||)
    sc->scTcb->tcbPriority < NODE_STATE(ksCurThread)->tcbPriority) {
      tcbSchedDequeue(sc->scTcb);
      …
```

No need to be syntactically correct

```
if (SMP_COND_STATEMENT(sc->scCore != getCurrentCPUIndex() ||)
    sc->scTcb->tcbPriority < NODE_STATE(ksCurThread)->tcbPriority) {
      tcbSchedDequeue(sc->scTcb);
      …
```

No need to be syntactically correct

```
if (SMP_COND_STATEMENT(sc->scCore != getCurrentCPUIndex() ||)
    sc->scTcb->tcbPriority < NODE_STATE(ksCurThread)->tcbPriority) {
      tcbSchedDequeue(sc->scTcb);
      …
```

```
#ifdef ENABLE_SMP_SUPPORT
#define SMP_COND_STATEMENT(_st)  _st
#else
#define SMP_COND_STATEMENT(_st)
#endif
```

Can’t we avoid all that?

Terrible semantics by text replacement in 2025.

Can’t we avoid all that somehow?

Can’t we avoid all that?

Terrible semantics by text replacement in 2025.

Can’t we avoid all that somehow?

No, not in C

(But please don’t put token-level macros into new languages)

Can’t we avoid all that?

Terrible semantics by text replacement in 2025.

Can’t we avoid all that somehow?

(But pleas

nguages)

Well, somewhat

(rest of this talk)

Classes of config options

Only attempt to treat the cases we use:

- Numbers
- Options/Booleans

4 Numbers

Abstracting over numbers

Easy in theory

Define a constant.

Assume/derive properties.

Don't unfold the constant.

Done.

Abstracting over numbers

Easy in theory

Define a constant.

Assume/derive properties.

Don't unfold the constant.

Done.

In practice

```
if (unlikely(hw_irq > maxIRQ) || …
```

Abstracting over numbers

Easy in theory

Define a constant.

Assume/derive properties.

Don’t unfold the constant.

Done.

In practice

```
if (unlikely(hw_irq > 187) || …
```

Force a symbolic name in C

read-only global

enum

static inline function

```
const word_t maxIRQ = 187;
```

```
enum IRQConstants {
    maxIRQ = 187
};
```

```
static inline word_t maxIRQ(void) {
  return 187;
}
```

Force a symbolic name in C

Simple and easy,
but compiler may not optimise or
use constant folding

read-only global

enum

static inline function

```
enum IRQConstants {
    maxIRQ = 187
};
```

```
static inline word_t maxIRQ(void) {
  return 187;
}
```

Force a symbolic name in C

read-only global

enum

static inline function

Simple and easy,
but compiler may not optimise or
use constant folding

only works for `int`

```
static inline word_t maxIRQ(void) {
  return 187;
}
```

Force a symbolic name in C

read-only global

enum

static inline function

Simple and easy,
but compiler may not optimise or
use constant folding

only works for `int`

Not idiomatic C, but works.
Compiler will treat it mostly like a
literal number.

```
oid) {
```

Numbers in types

```
irq_state_t intStateIRQTable[maxIRQ+1];
```

Numbers in types

```
irq_state_t intStateIRQTable[187+1];
```

Numbers in types

```
irq_state_t intStateIRQTable[187+1];
```

188 as a numeral type is Ok in Isabelle

```
type_synonym irqTable = irqState[188]
```

Numbers in types

```
irq_state_t intStateIRQTable[187+1];
```

188 as a numeral type is Ok in Isabelle

```
type_synonym irqTable = irqState[188]
```

```
type_synonym irqTable = irqState[maxIRQ+1]
```

Numbers in types

```
irq_state_t intStateIRQTable[187+1];
```

188 as a numeral type is Ok in Isabelle

```
type_synonym irqTable = irqState[188]
```

`maxIRQ+1` is a term, not a type

~~`type_synonym irqTable = irqState[maxIRQ+1]`~~

New Isabelle command value_type

```
value_type irq_sz = maxIRQ + 1
```

New Isabelle command value_type

```
value_type irq_sz = maxIRQ + 1
```

value_type ty = t

- evaluates right-hand term `t` to a number
- declares a type synonym `ty` = number
- proves `CARD(ty) = t`

New Isabelle command value_type

```
value_type irq_sz = maxIRQ + 1
```

value_type ty = t

- evaluates right-hand term `t` to a number
- declares a type synonym `ty` = number
- proves `CARD(ty) = t`

```
type_synonym irqTable = irqState[irq_sz]
```

4 Boolean Options

Abstracting over boolean options

Also easy in theory

Define a constant.

Use `if-then-else`

Don't unfold the constant,
make a case distinction instead.

Done.

Abstracting over boolean options

Also easy in theory

Define a constant.

Use `if-then-else`

Don't unfold the constant,
make a case distinction instead.

One proof for both cases

Abstracting over boolean options

Also easy in theory

Define a constant.

Use `if-then-else`

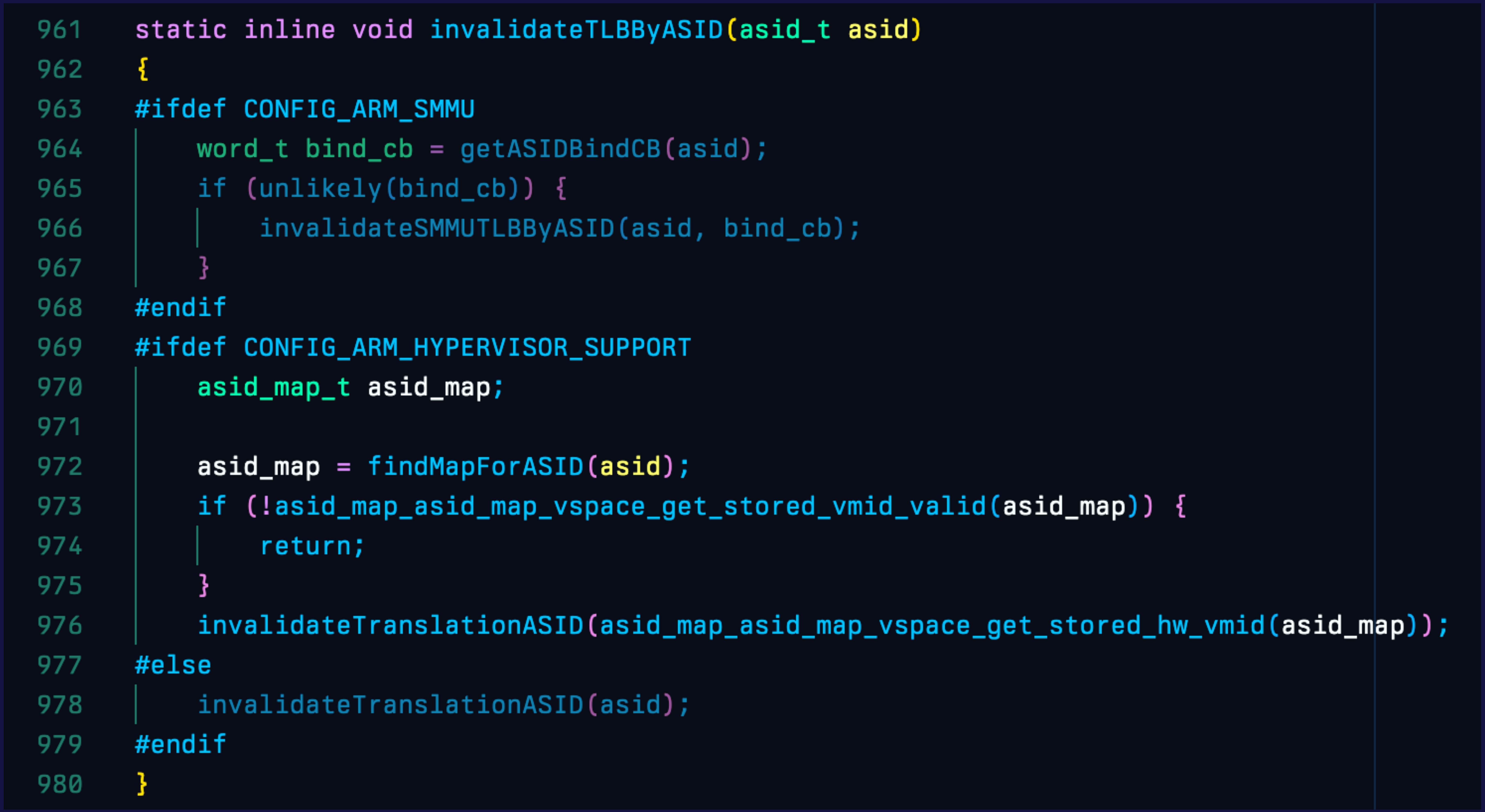
Don't unfold the constant,
make a case distinction instead.

One proof for both cases

In practice

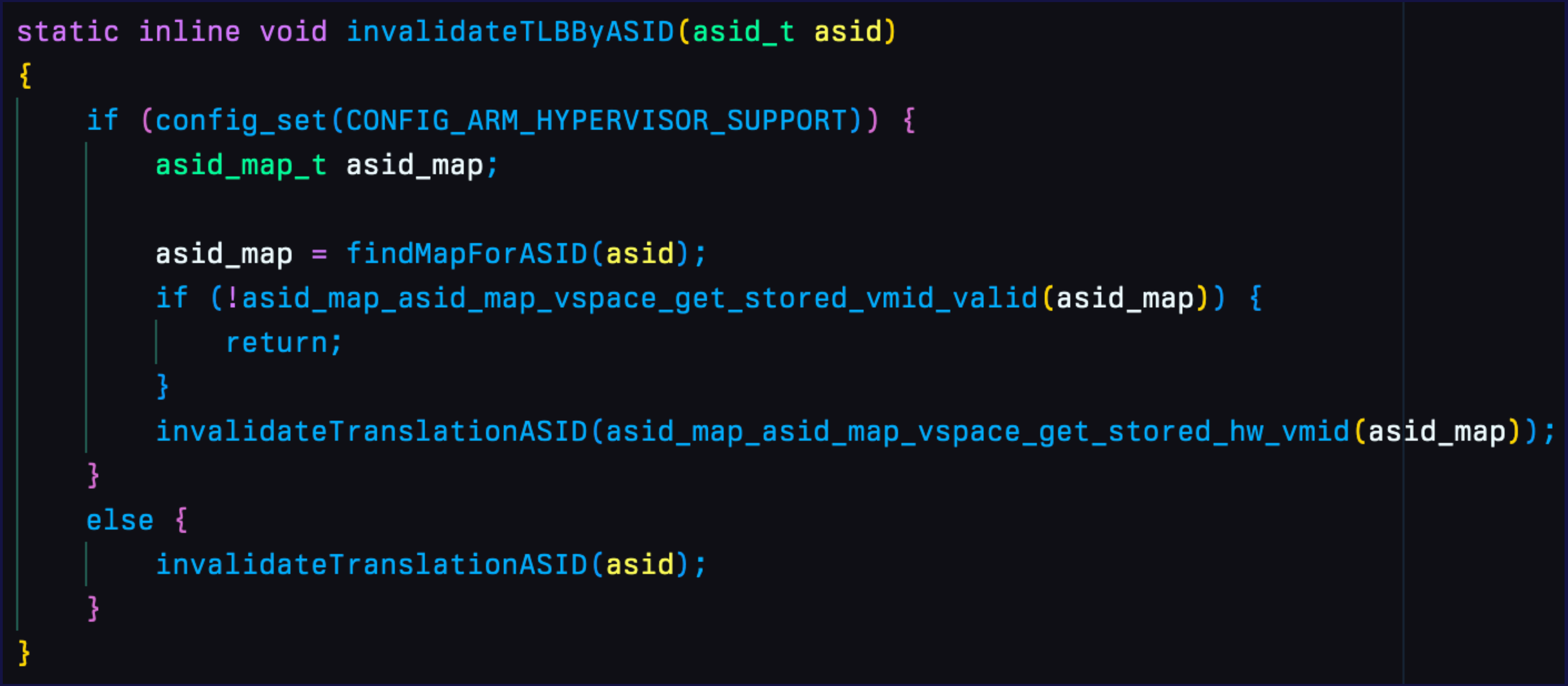
Config options are used to eliminate code and change shape of data structures

#ifdef changes code shape



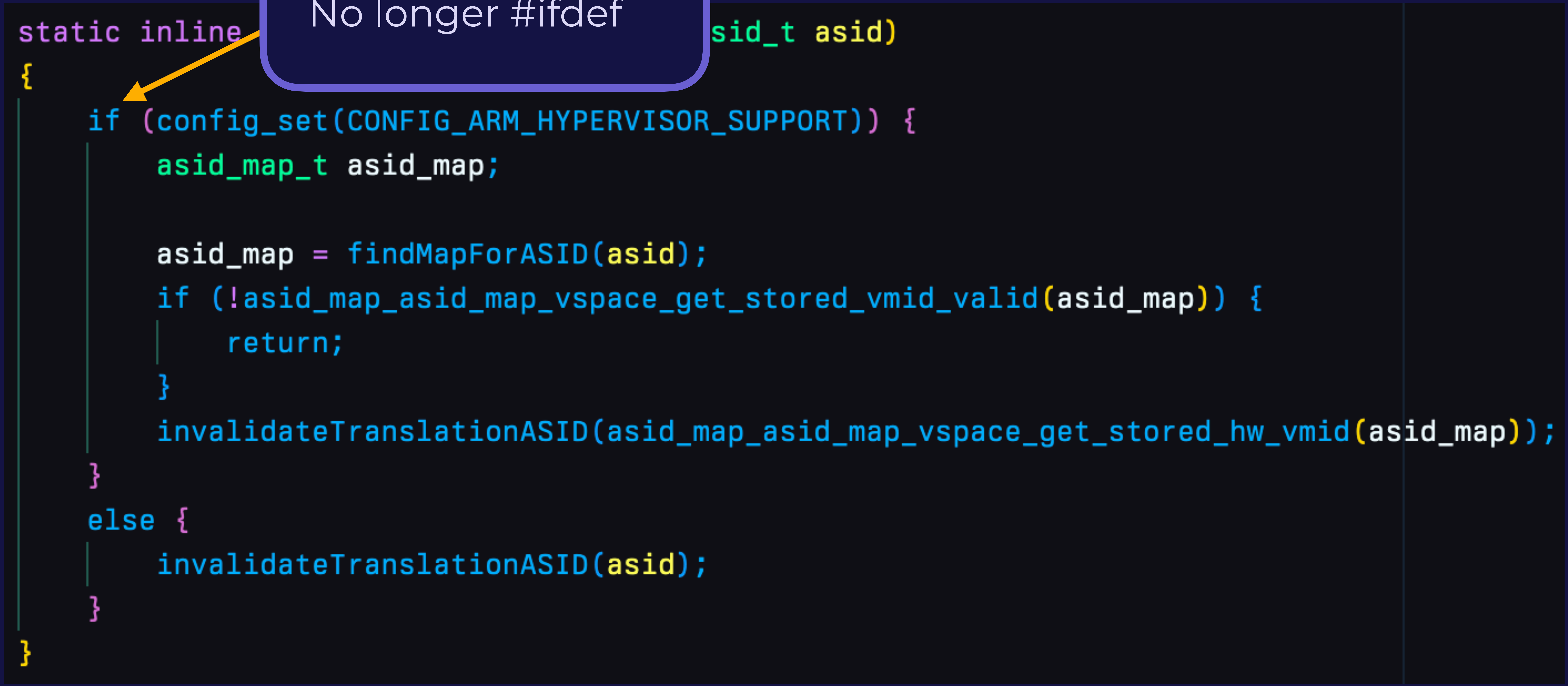
```
static inline void invalidateTLBByASID(asid_t asid)
{
#ifdef CONFIG_ARM_SMMU
    word_t bind_cb = getASIDBindCB(asid);
    if (unlikely(bind_cb)) {
        invalidateSMMUTLBByASID(asid, bind_cb);
    }
#endif
#ifdef CONFIG_ARM_HYPERVISOR_SUPPORT
    asid_map_t asid_map;

    asid_map = findMapForASID(asid);
    if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
        return;
    }
    invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
#else
    invalidateTranslationASID(asid);
#endif
}
```

#ifdef -> if



```
static inline void invalidateTLBByASID(asid_t asid)
{
    if (config_set(CONFIG_ARM_HYPERVISOR_SUPPORT)) {
        asid_map_t asid_map;

        asid_map = findMapForASID(asid);
        if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
            return;
        }
        invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
    }
    else {
        invalidateTranslationASID(asid);
    }
}
```

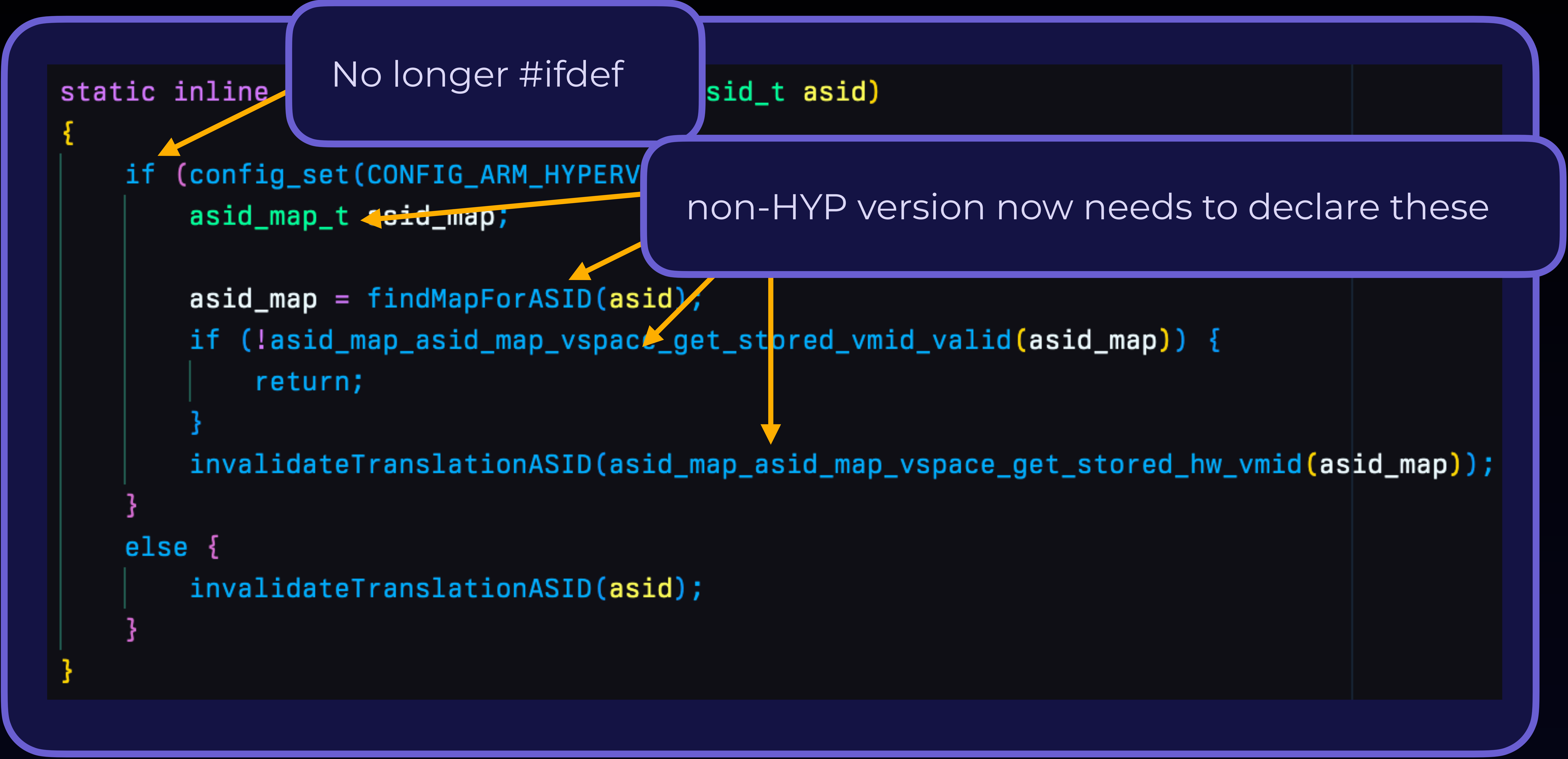
#ifdef -> if



No longer #ifdef

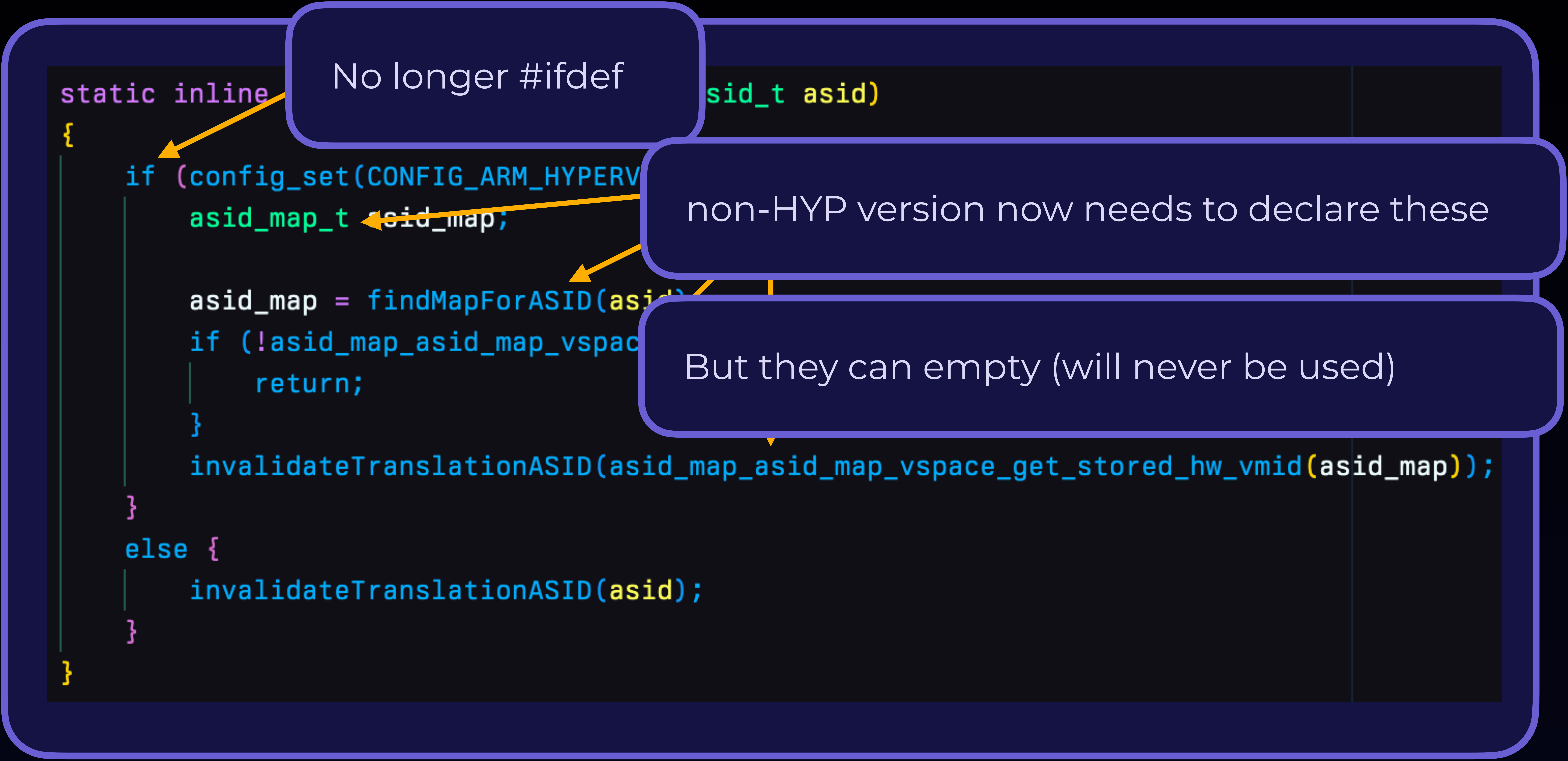
```c
static inline                                       sid_t asid)
{
    if (config_set(CONFIG_ARM_HYPERVISOR_SUPPORT)) {
        asid_map_t asid_map;

        asid_map = findMapForASID(asid);
        if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
            return;
        }
        invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
    }
    else {
        invalidateTranslationASID(asid);
    }
}
```

#ifdef -> if



```
static inline                               sid_t asid)
{
    if (config_set(CONFIG_ARM_HYPERV
        asid_map_t asid_map;

        asid_map = findMapForASID(asid);
        if (!asid_map_asid_map_vspace_get_stored_vmid_valid(asid_map)) {
            return;
        }
        invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
    }
    else {
        invalidateTranslationASID(asid);
    }
}
```

#ifdef -> if



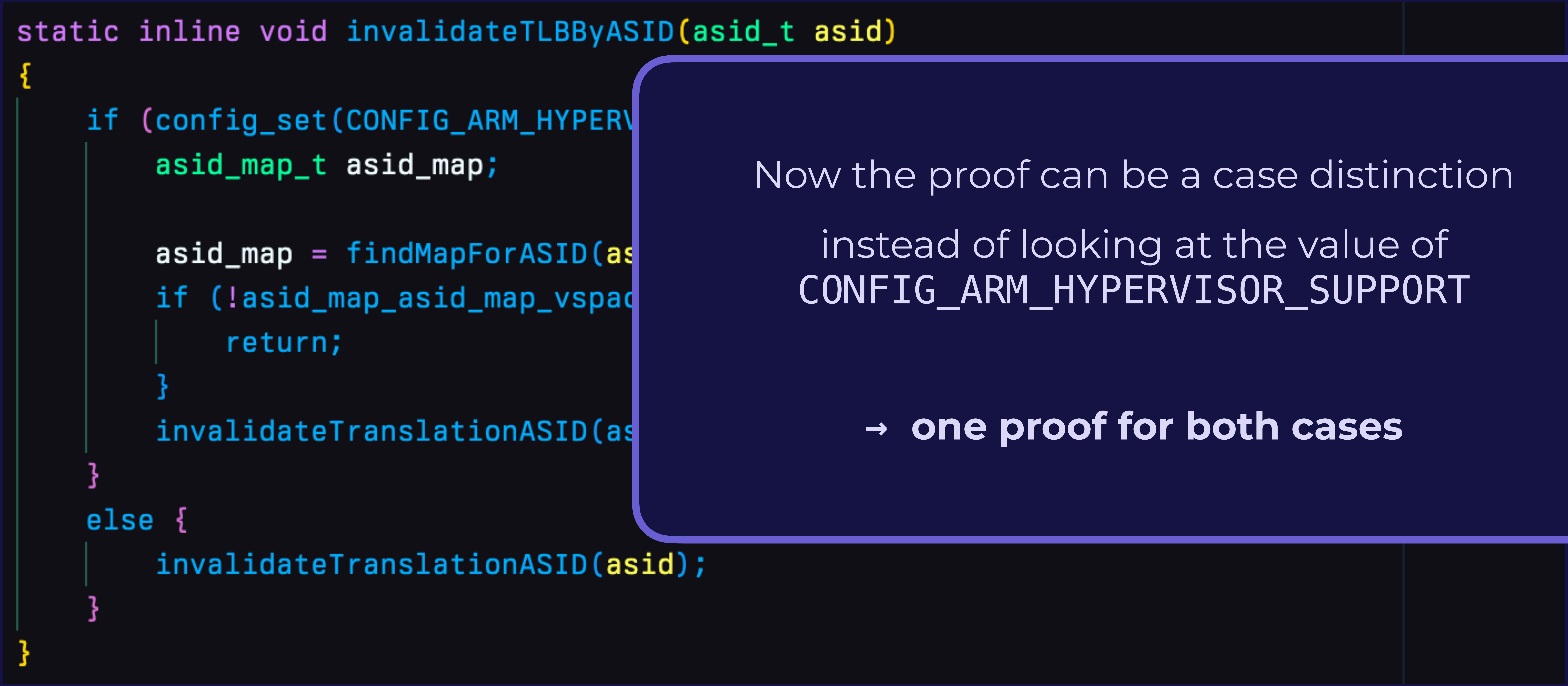
```
static inline                                    sid_t asid)
{
    if (config_set(CONFIG_ARM_HYPERV
        asid_map_t asid_map;

        asid_map = findMapForASID(asid);
        if (!asid_map_asid_map_vspac
            return;
        }
        invalidateTranslationASID(asid_map_asid_map_vspace_get_stored_hw_vmid(asid_map));
    }
    else {
        invalidateTranslationASID(asid);
    }
}
```

No longer #ifdef

non-HYP version now needs to declare these

But they can empty (will never be used)

#ifdef -> if



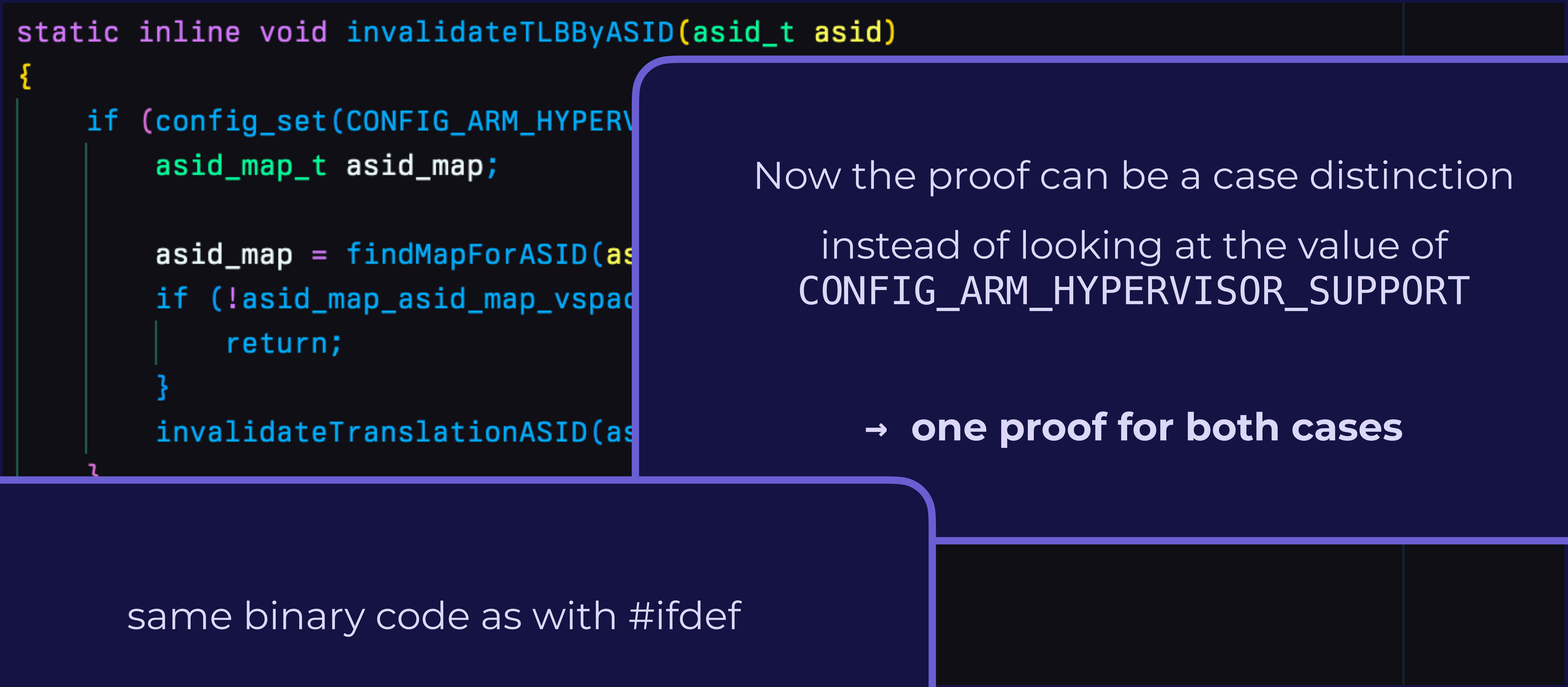
```
static inline void invalidateTLBByASID(asid_t asid)
{
    if (config_set(CONFIG_ARM_HYPERV
        asid_map_t asid_map;

        asid_map = findMapForASID(as
        if (!asid_map_asid_map_vspac
            return;
        }
        invalidateTranslationASID(as
    }
    else {
        invalidateTranslationASID(asid);
    }
}
```

Now the proof can be a case distinction

instead of looking at the value of `CONFIG_ARM_HYPERVISOR_SUPPORT`

→ **one proof for both cases**

#ifdef -> if



```
static inline void invalidateTLBByASID(asid_t asid)
{
    if (config_set(CONFIG_ARM_HYPERV
        asid_map_t asid_map;

        asid_map = findMapForASID(as
        if (!asid_map_asid_map_vspac
            return;
        }
        invalidateTranslationASID(as
    }
```

Now the proof can be a case distinction

instead of looking at the value of `CONFIG_ARM_HYPERVISOR_SUPPORT`

→ **one proof for both cases**

same binary code as with #ifdef

4

Summary

Verified seL4 platforms: currently

- Arm:
  - ✓Avnet MaaXBoard
  - ✓BeagleBoard
  - ✓BeagleBone Black
  - ✓IMX8MM-EVK
  - ✓Odroid-C2
  - ✓Odroid-C4
  - ✓OdroidX
  - ✓OdroidXU
  - ✓OdroidXU4
  - ✓Raspberry Pi 3b
  - ✓Raspberry Pi 4B
  - ✓Rockpro64
  - ✓Sabre Lite
  - ✓TK1
  - ✓TK1-SOM
  - ✓TX1
  - ✓TX2
  - ✓Ultra96v2
  - ✓Zynq ZCU102
  - ✓Zynq-7000
  - ✓IMX8MQ
  - ✓ZCU106
- New: ✓IMX93
- RISC-V:
  - Ariane
  - Cheshire
  - ✓HiFive Unleashed
  - Microchip PolarFire
  - Rocketchip
- Intel:
  - 32 bit PC-99
  - ✓64 bit PC-99

Verified seL4 platforms: currently

- Arm:
  - ✓Avnet MaaXBoard
  - ✓BeagleBoard
  - ✓BeagleBone Black
  - ✓IMX8MM-EVK
  - ✓Odroid-C2
  - ✓Odroid-C4
  - ✓OdroidX
  - ✓Rockpro64
  - ✓Sabre Lite
  - ✓TK1
  - ✓TK1-SOM
  - ✓TX1
  - ✓TX2
  - ✓Ultra96v2
- RISC-V:
  - Ariane
  - Cheshire
  - ✓HiFive Unleashed
  - Microchip PolarFire
  - Rocketchip

100% of seL4 Arm platforms now verified

More to come: 1,099,511,627,776 kernels looks achievable

What else to expect

Also happening in seL4 verification

- **AArch64** integrity proof completed,
  confidentiality proof next
- **MCS** proofs in progress
- **Multikernel** proofs in progress

Thank You