Zephyr® Project
Developer Summit

Zephyr® Project
Developer Summit

Off-Chip Peripheral Emulators

Aaron E. Massey, Google
aaronmassey@google.com

aaronmassey_google aaronemassey

#EMBEDDEDOSSUMMIT

Zephyr® Project
Developer Summit

Goal:

Scalably Verify Peripheral Drivers

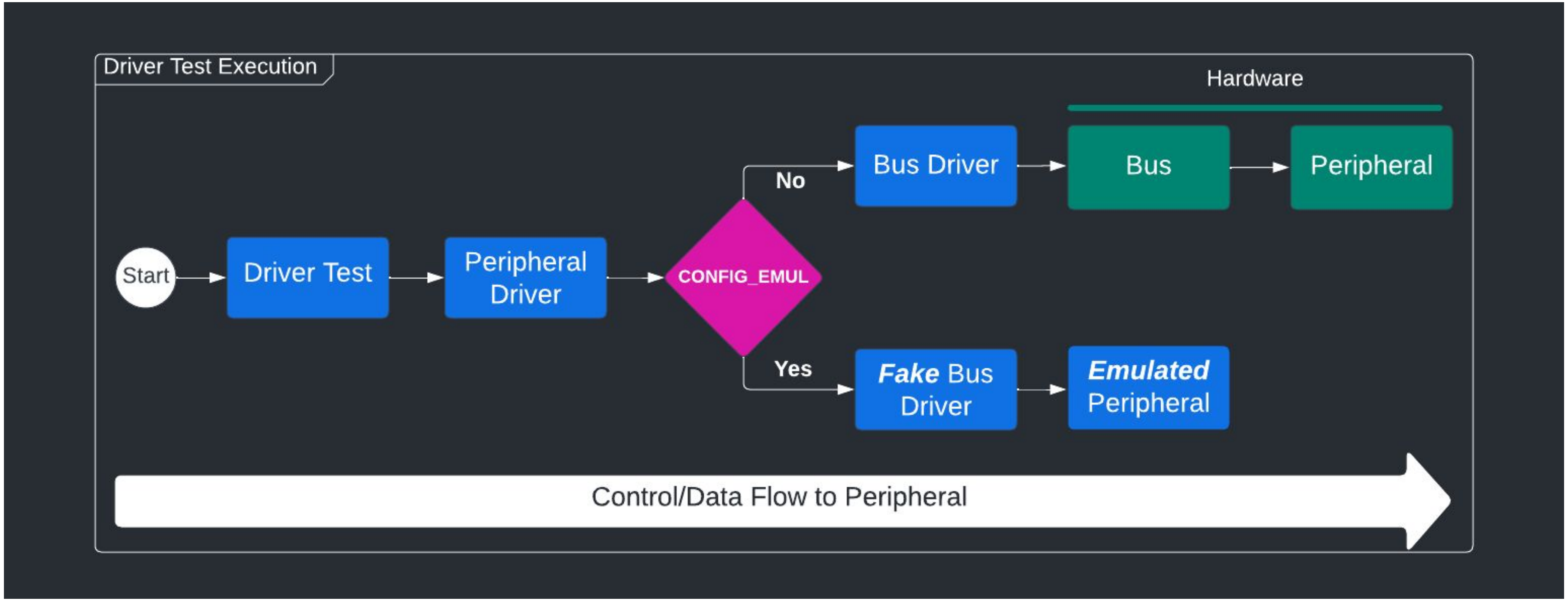
Agenda

- What is an emulator (peripheral emulator)?
- When and why should we emulate?
- Creating an emulator
- Using emulators
- Q&A

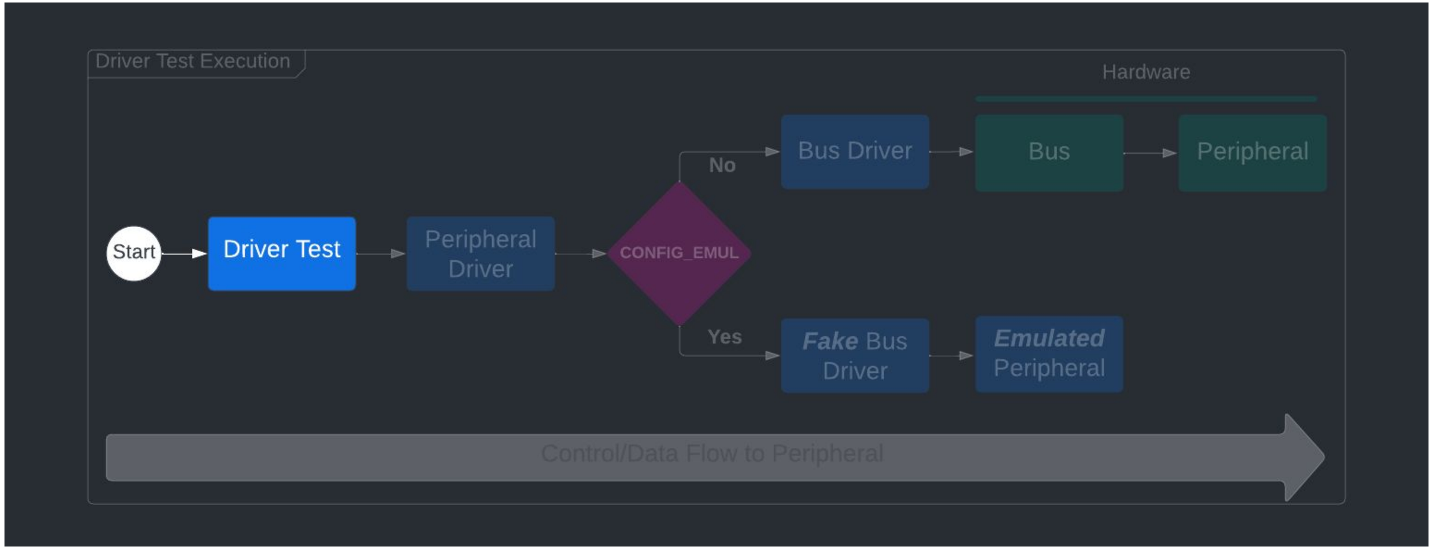
Agenda

- ***What is an emulator (peripheral emulator)?***
- When and why should we emulate?
- Creating an emulator
- Using emulators
- Q&A

Emulators - Faking Peripherals on a Bus

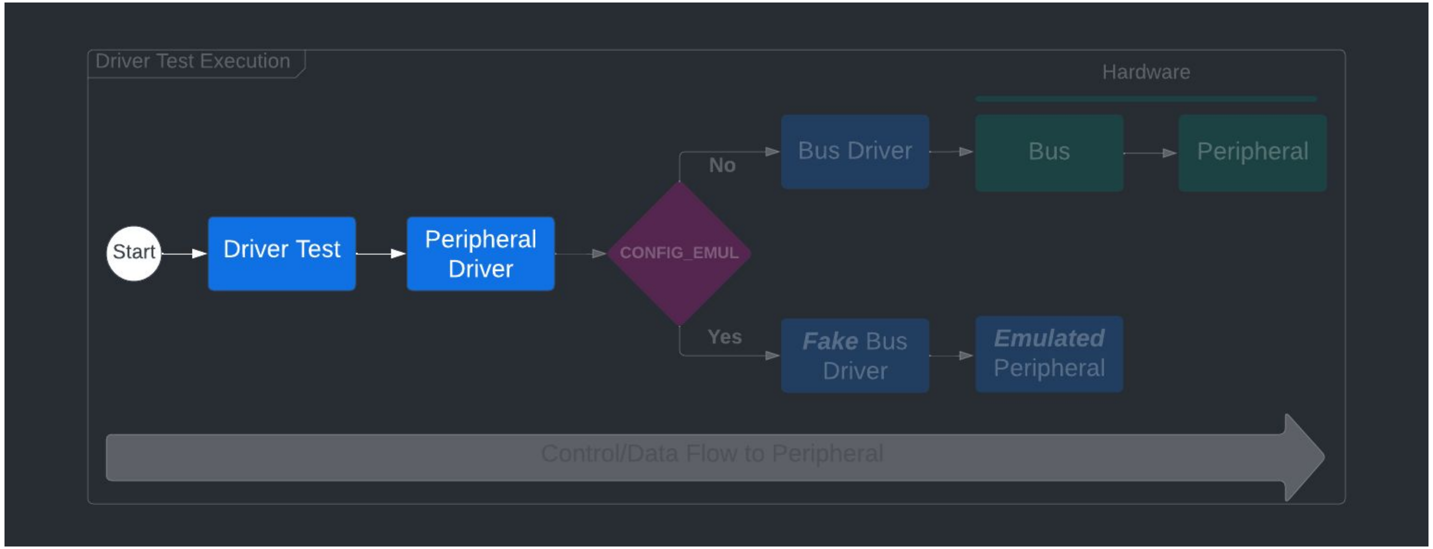


Emulators - Faking Peripherals on a Bus



```
// Test invoking API
fuel_gauge_get_prop(...)
```

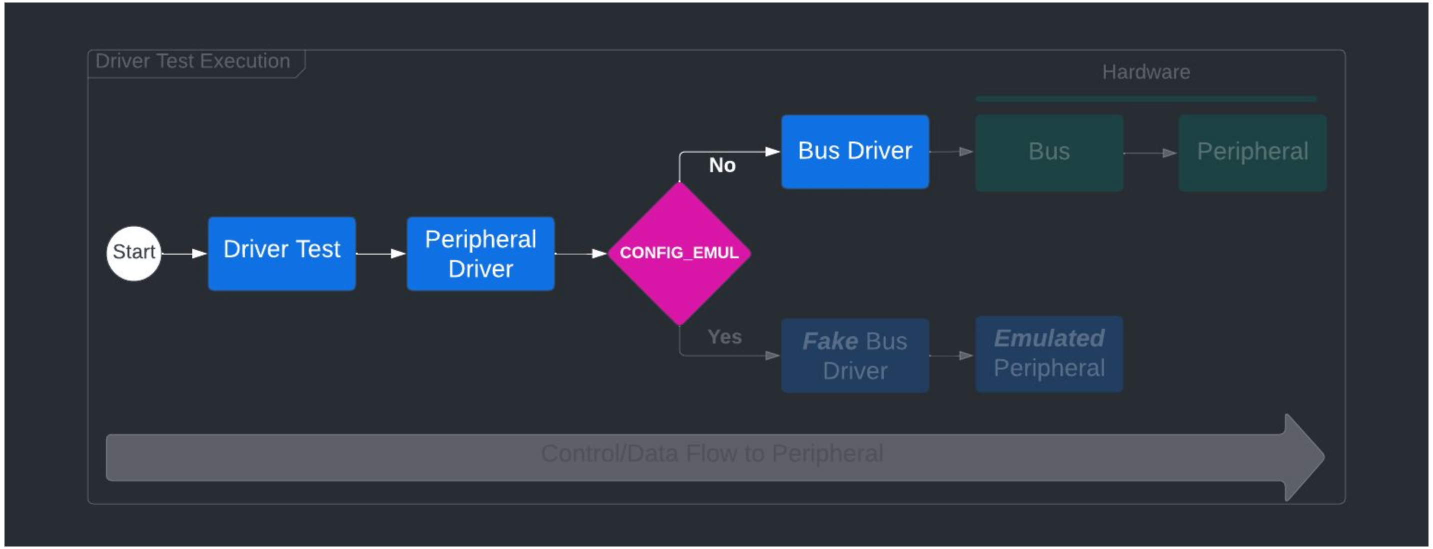
Emulators - Faking Peripherals on a Bus

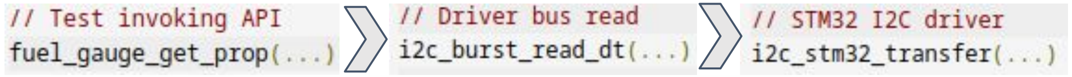


```
// Test invoking API
fuel_gauge_get_prop(...)
```

```
// Driver bus read
i2c_burst_read_dt(...)
```

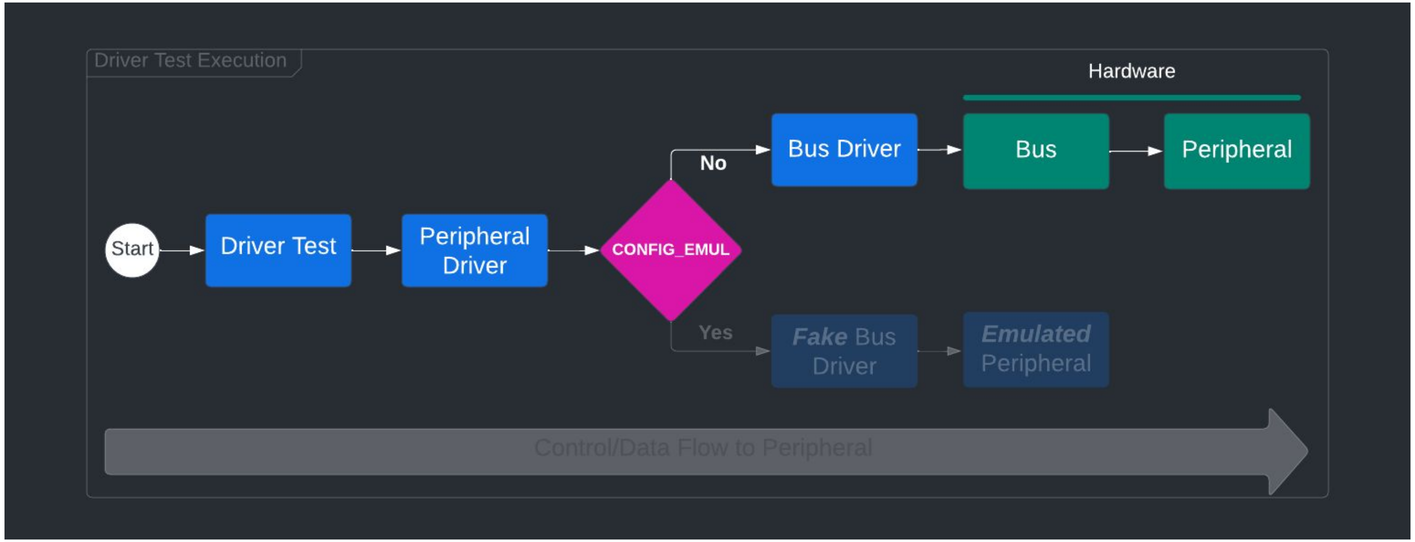
Emulators - Faking Peripherals on a Bus




```
// Test invoking API
fuel_gauge_get_prop(...)
```
```
// Driver bus read
i2c_burst_read_dt(...)
```
```
// STM32 I2C driver
i2c_stm32_transfer(...)
```

Zephyr® Project
Developer Summit

Emulators - Faking Peripherals on a Bus



```
// Test invoking API
fuel_gauge_get_prop(...)
```

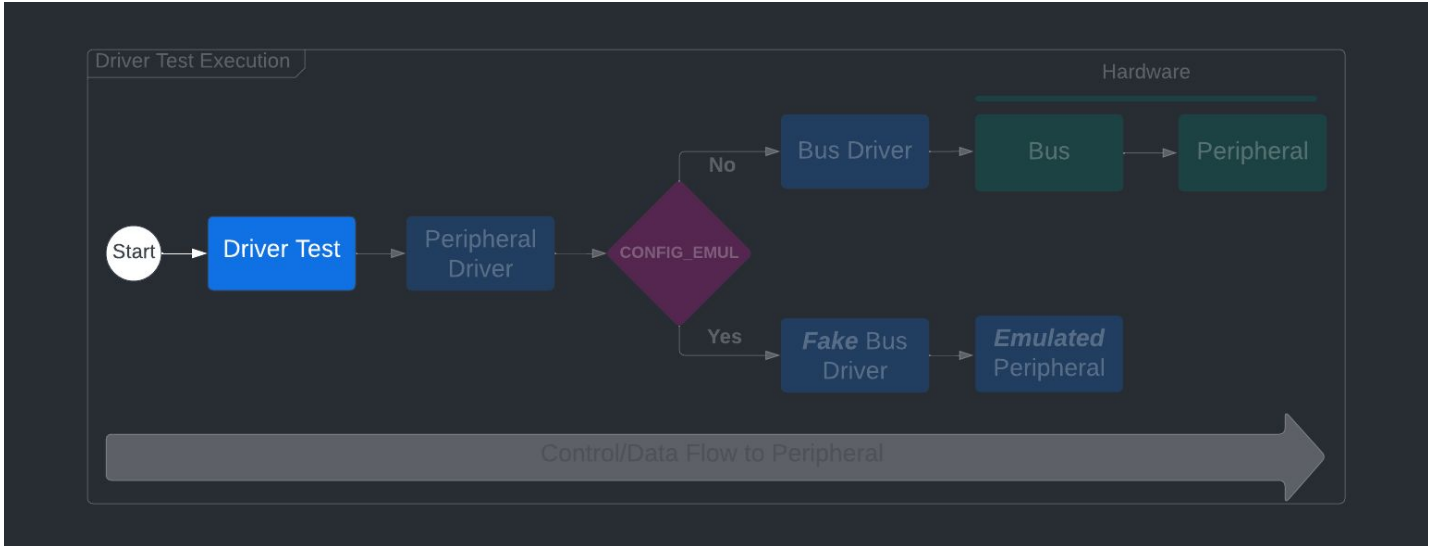
```
// Driver bus read
i2c_burst_read_dt(...)
```

```
// STM32 I2C driver
i2c_stm32_transfer(...)
```

Hardware

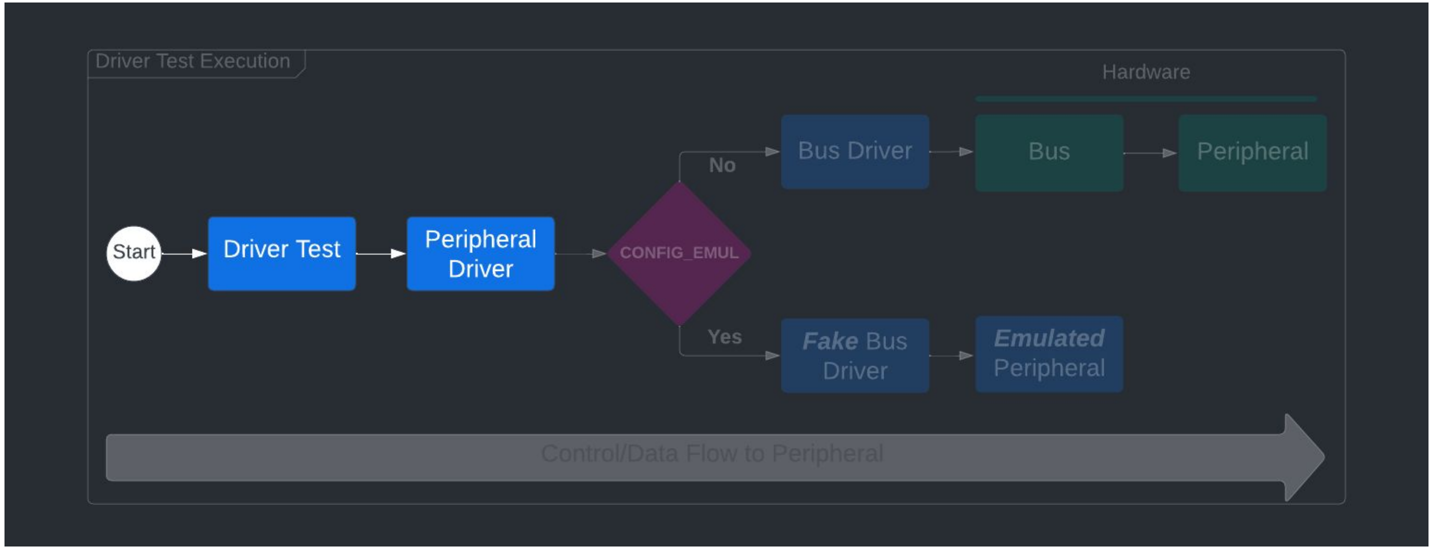
Zephyr® Project
Developer Summit

Emulators - Faking Peripherals on a Bus



```
// Test invoking API
fuel_gauge_get_prop(...)
```

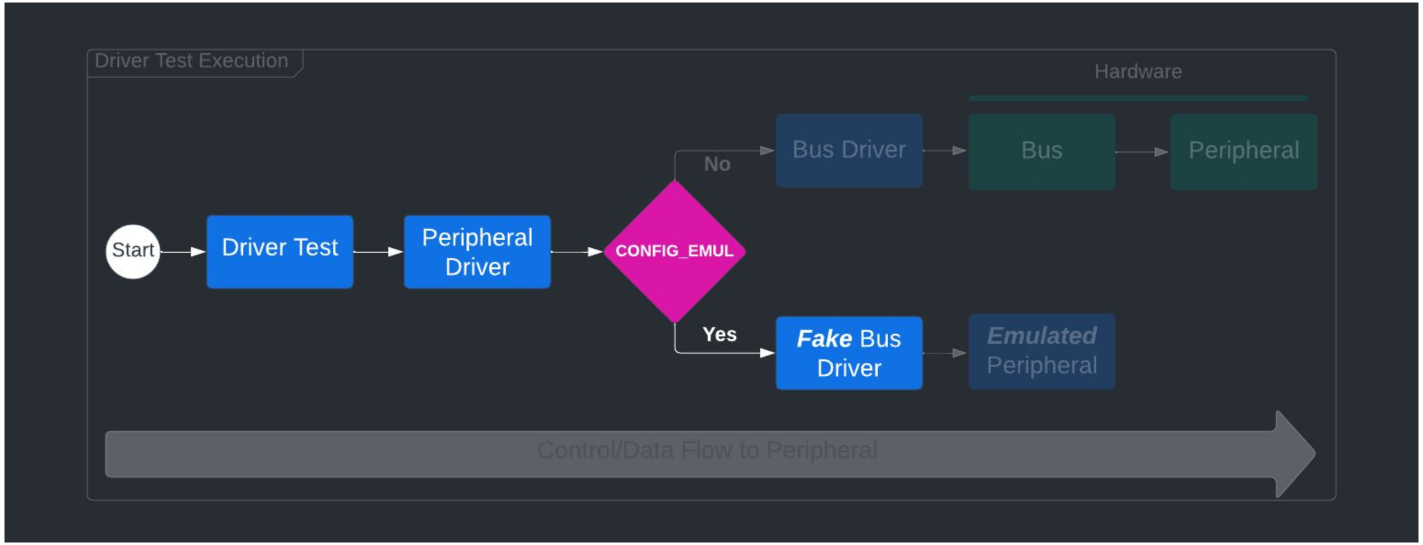
Emulators - Faking Peripherals on a Bus



```
// Test invoking API
fuel_gauge_get_prop(...)
```

```
// Driver bus read
i2c_burst_read_dt(...)
```

Emulators - Faking Peripherals on a Bus



```
// Test invoking API
fuel_gauge_get_prop(...)
```

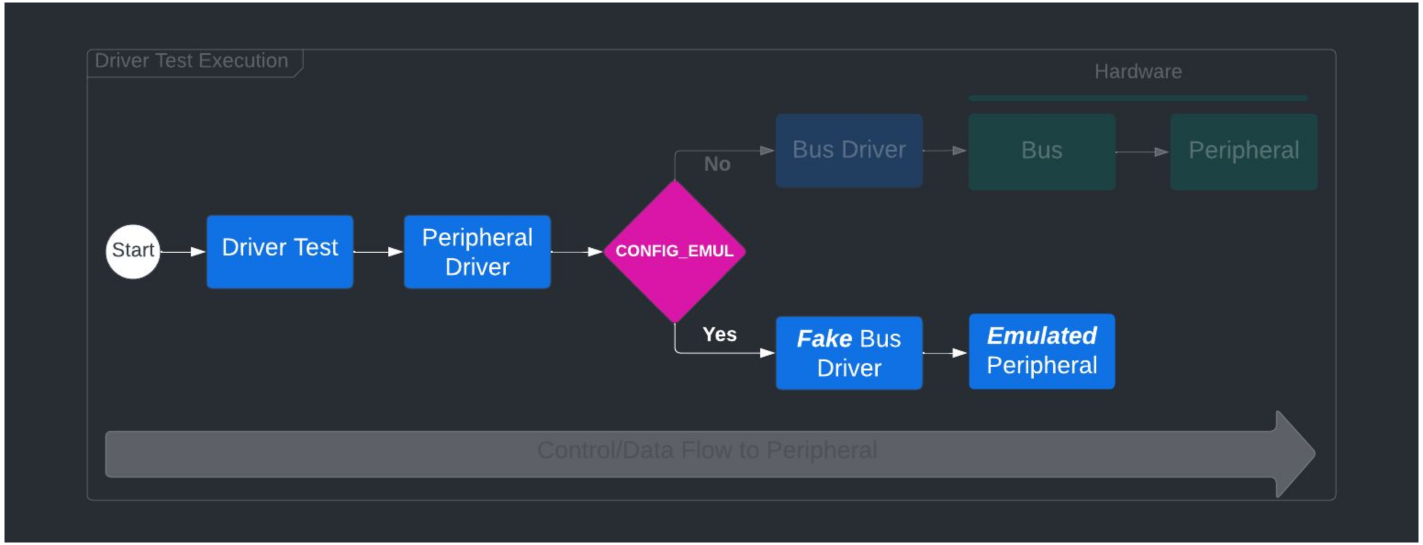
❯

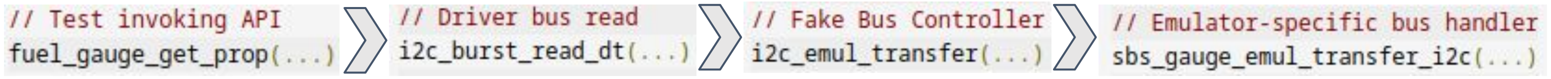
```
// Driver bus read
i2c_burst_read_dt(...)
```

❯

```
// Fake Bus Controller
i2c_emul_transfer(...)
```

Emulators - Faking Peripherals on a Bus





```
// Test invoking API
fuel_gauge_get_prop(...)
```

```
// Driver bus read
i2c_burst_read_dt(...)
```

```
// Fake Bus Controller
i2c_emul_transfer(...)
```

```
// Emulator-specific bus handler
sbs_gauge_emul_transfer_i2c(...)
```

Agenda

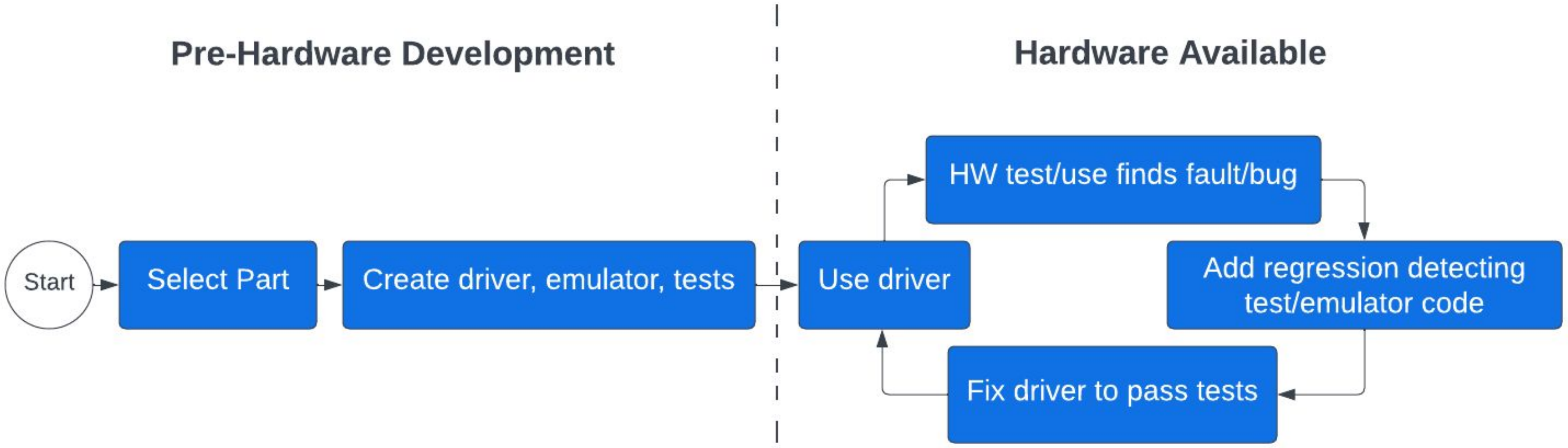
- What is an emulator (peripheral emulator)?
- ***When and why should we emulate?***
- Creating an emulator
- Using emulators
- Q&A

Why should we emulate? Good/Bad

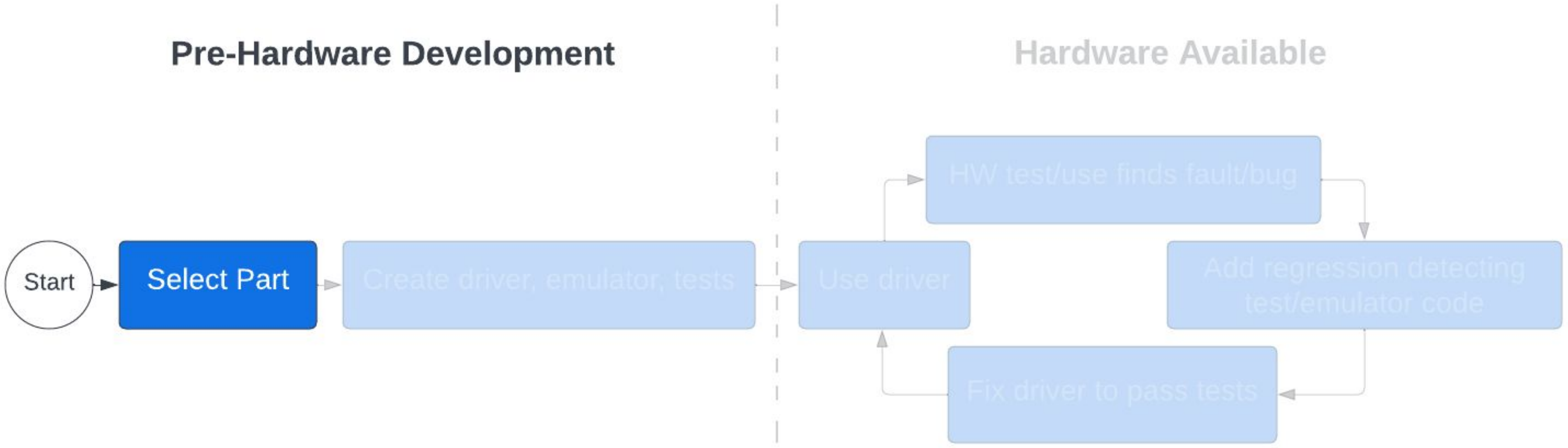
Emulated Testing	**Hardware Testing**
Hardware-less	Requires hardware
Scalable & stable - Zephyr CI	Slower & more non-deterministic
Debug on developer machine	Debug on test device
Easy to recreate test scenario	Difficult to recreate test scenario
Emulated	**Authentic**
Control + velocity over **accuracy**	**Accuracy** over **control + velocity**

Testing is better with both!

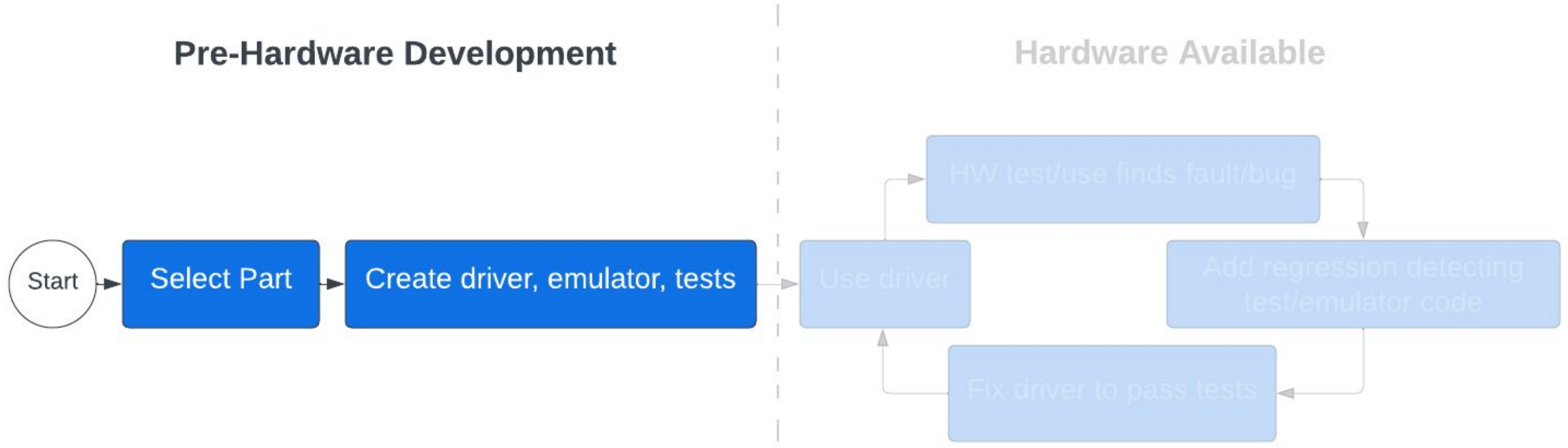
When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

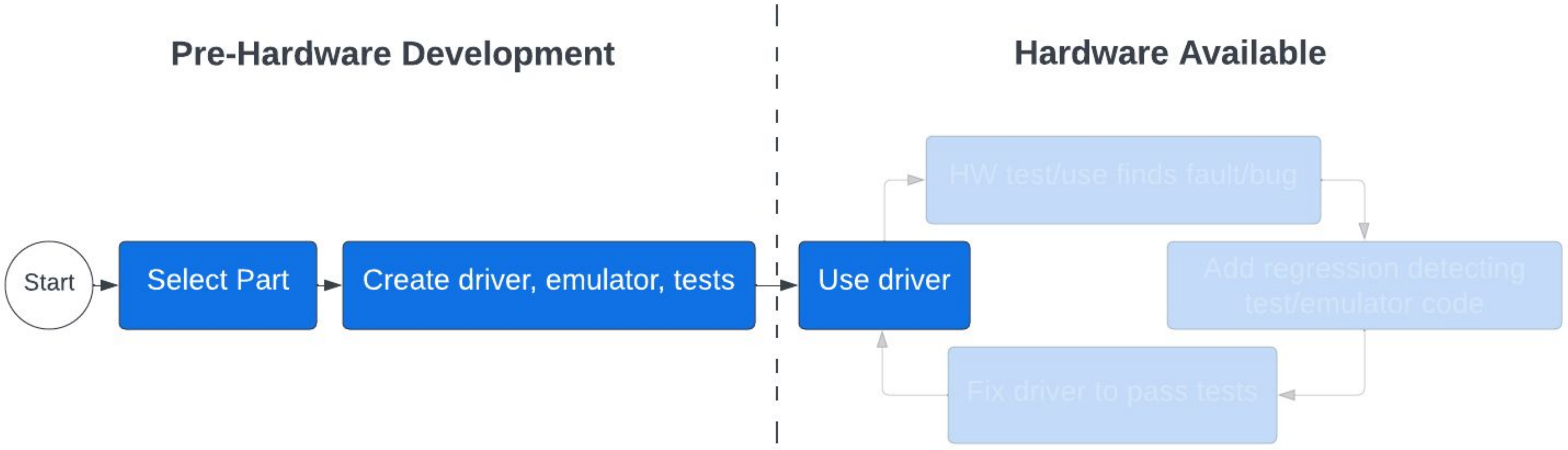
When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

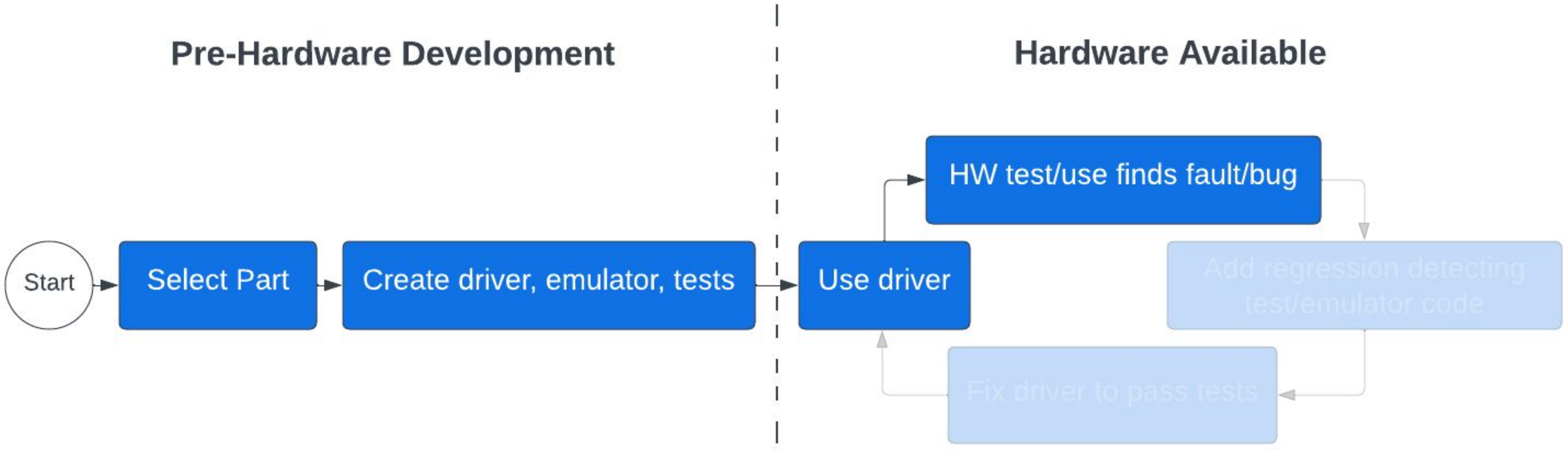
When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

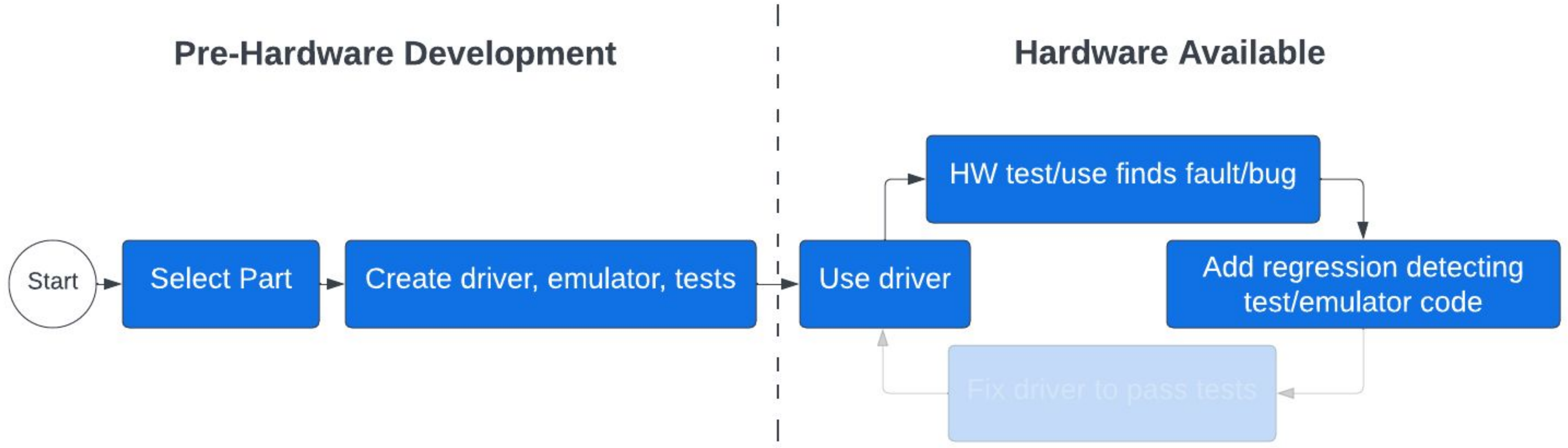
When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

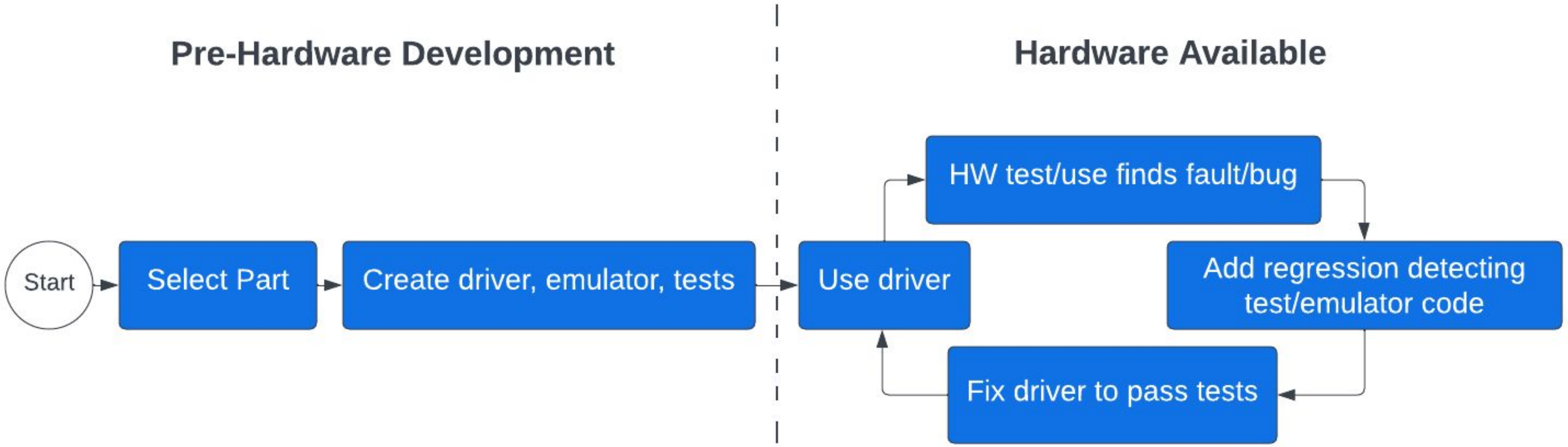
When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

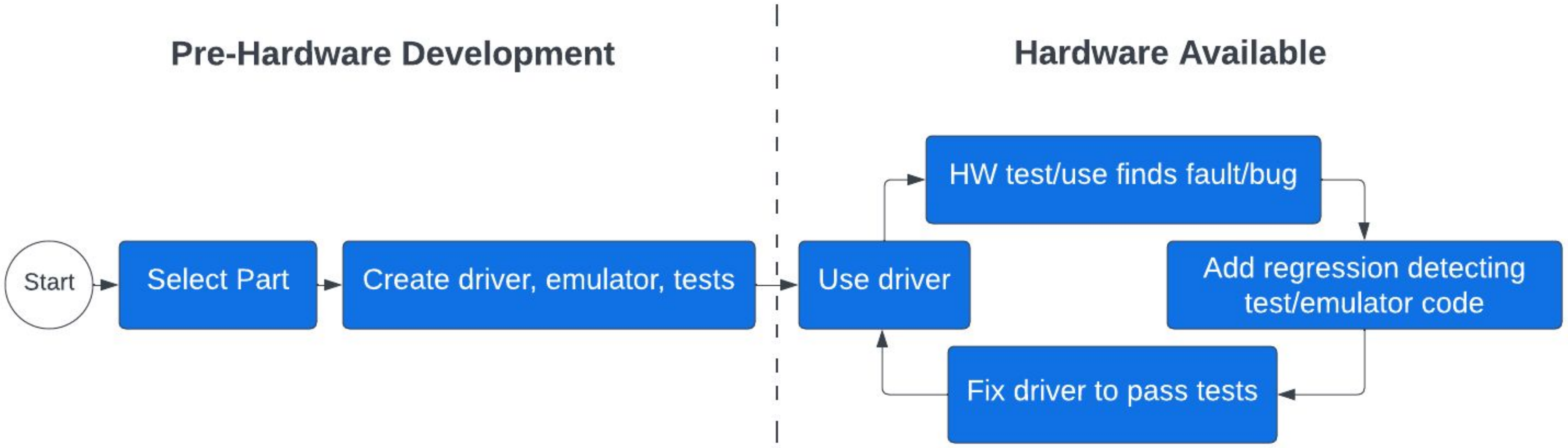
When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

When should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

When should we emulate?

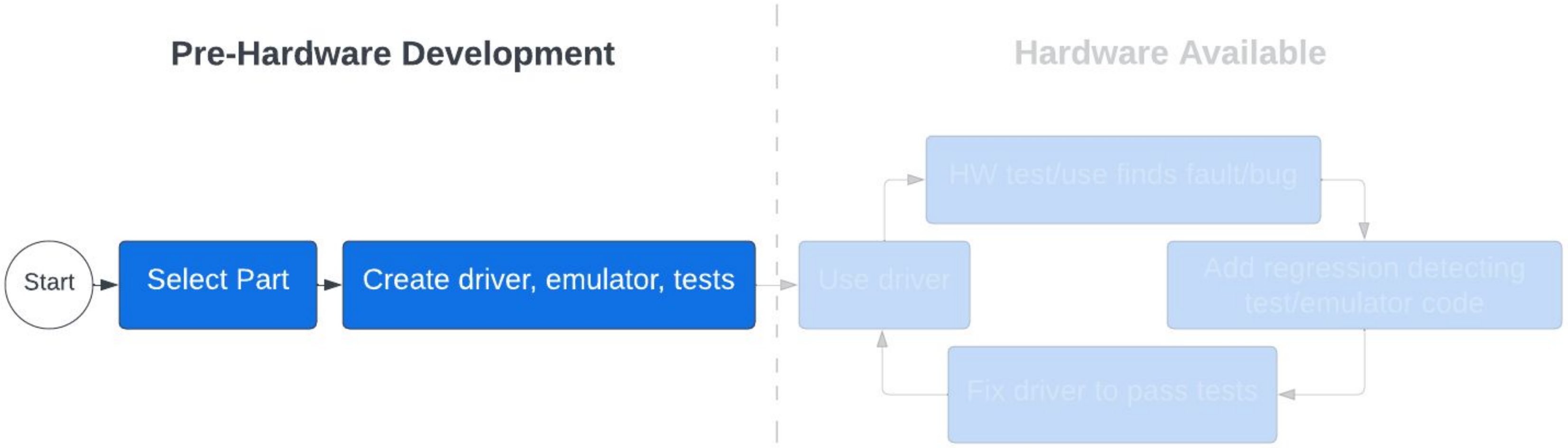

Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

Test Driven Development!

Agenda

- What is an emulator (peripheral emulator)?
- When and why should we emulate?
- ***Creating an emulator***
- Using emulators
- Q&A

When and why should we emulate?


Pre-Hardware Development
Hardware Available
Start
Select Part
Create driver, emulator, tests
Use driver
HW test/use finds fault/bug
Add regression detecting test/emulator code
Fix driver to pass tests

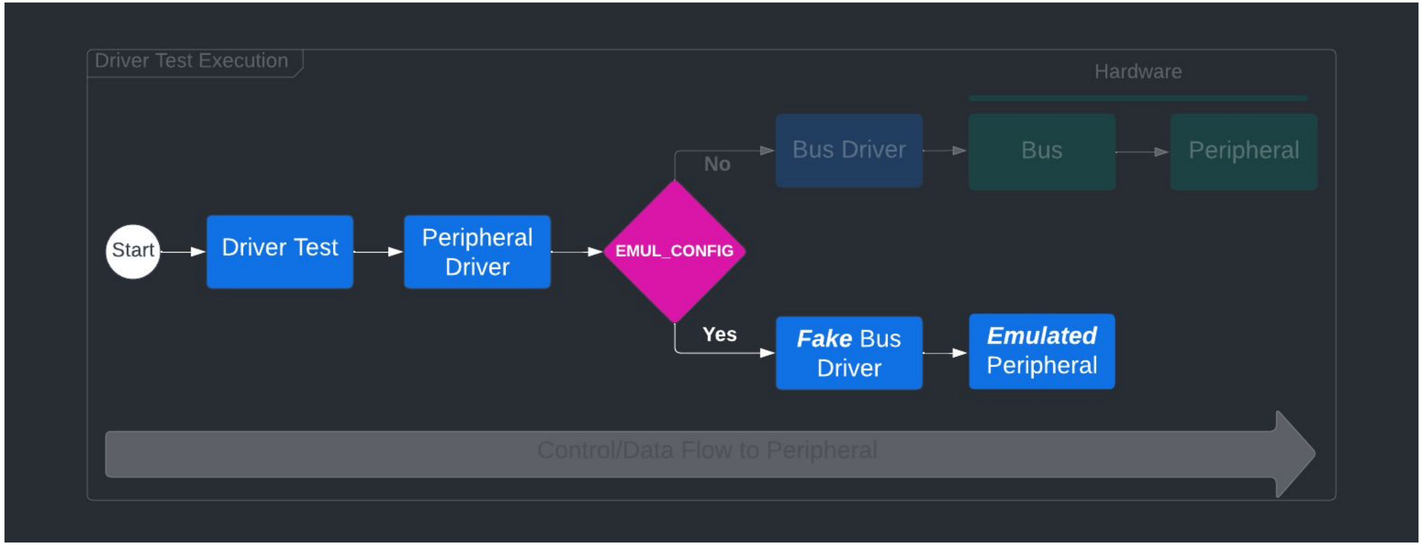
Creating an emulator

- Emulators typically consist of a single C source.
- Similar to defining a device driver

```
DEVICE_DT_DEFINE(node_id, init_fn, pm, data, config, level, prio, api)

EMUL_DT_DEFINE(node_id, init_fn, data, cfg, bus_api, backend_api)
```

- Parameters specific to emulators
  - **bus_api - *bus messaging* (required)**
  - **backend_api** - *test scenario setup* (optional but useful)

Emulators - Faking Peripherals on a Bus



```
// Test invoking API
fuel_gauge_get_prop(...)
```
❯
```
// Driver bus read
i2c_burst_read_dt(...)
```
❯
```
// Fake Bus Controller
i2c_emul_transfer(...)
```
❯
```
// Emulator-specific bus handler
sbs_gauge_emul_transfer_i2c(...)
```

Creating an emulator - I2C *bus_api example*

```
akm09918c_emul_transfer_i2c(const struct emul *target,
                            struct i2c_msg *msgs,
                            int num_msgs, int addr)
{
    if (is_read) {
        /* handle register read */
    } else if (is_write) {
        /* handle register write */
    }
    else {
        /* handle unknown case */
    }
}
```

Creating an emulator

- Emulators typically consist of a single C source.
- Similar to defining a device driver

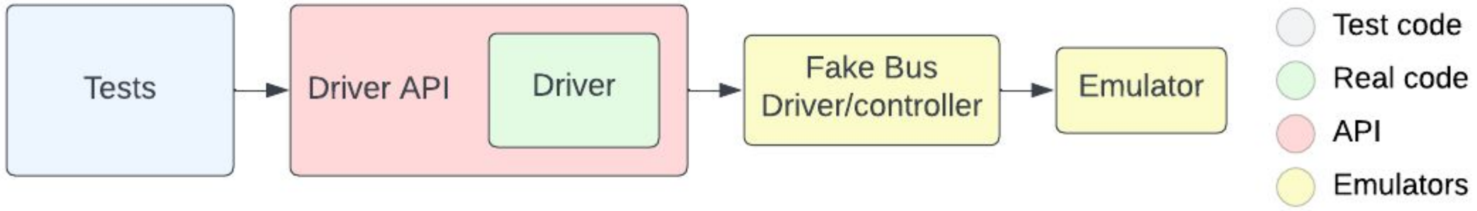
```
DEVICE_DT_DEFINE(node_id, init_fn, pm, data, config, level, prio, api)

EMUL_DT_DEFINE(node_id, init_fn, data, cfg, bus_api, backend_api)
```

- Parameters specific to emulators
  - **bus_api** - *bus messaging* (required) ✓
  - **backend_api - *test scenario setup* (optional but useful)**

Zephyr® Project
Developer Summit

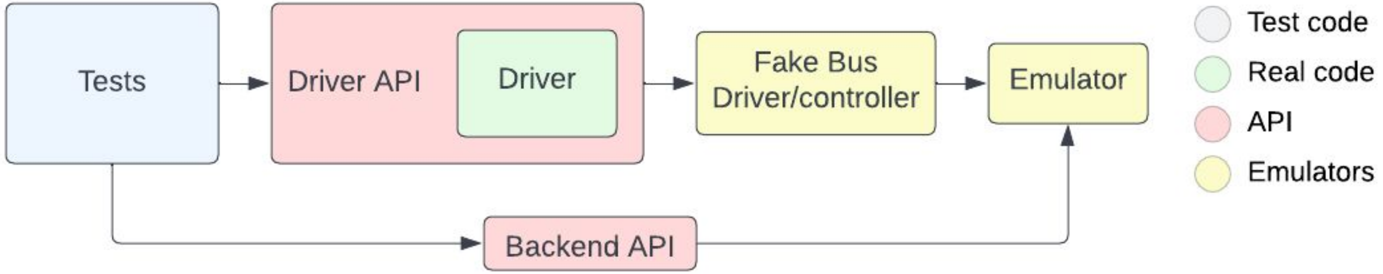
Creating an emulator - *backend_api*

- **Goal**: Test scenario setup



Creating an emulator - *backend_api*

- **Goal**: Test scenario setup
- **Solution**: Emulator provides an API



Creating an emulator - *backend_api - bc12 example*

```
#include <zephyr/drivers/usb/emul_bc12.h>
...
ZTEST_USER_F(bc12_pd_mode, test_bc12_sdp_charging_partner)
{
	/* Connect a SDP charging partner to the emulator */
	bc12_emul_set_charging_partner(fixture->bc12_emul, BC12_TYPE_SDP);
	...
	/* Verify bc12 driver set partner state in callback */
	zassert_equal(fixture->partner_state.type, BC12_TYPE_SDP);
}
```

Creating an emulator - *backend_api - bc12 example*

```
__subsystem struct bc12_emul_driver_api {

        int (*set_charging_partner)(const struct emul *emul,
                                    enum bc12_type partner_type);
};


int bc12_emul_set_charging_partner(const struct emul *target,
                                   enum bc12_type partner_type)
```

Creating an emulator - *backend_api - bc12 example*

```
int pi3usb9201_emul_set_charging_partner(const struct emul *target,
                                          enum bc12_type partner_type)
{
        struct pi3usb9201_emul_data *data = target->data;

        // If bad partner type fail

        // Otherwise modify internal registers to reflect charging partner
        // E.g. if partner_type is SDP:
        // data->test_client_status = SDP_DETECTED;
}
```

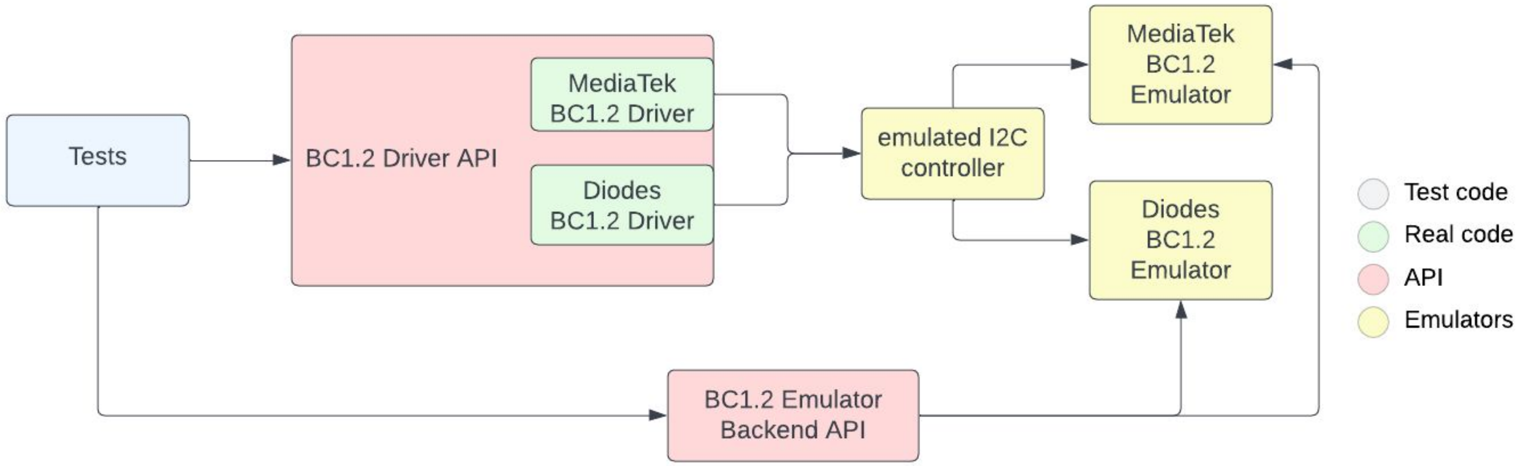
Creating an emulator

- Emulators typically consist of a single C source.
- Similar to defining a device driver

```
DEVICE_DT_DEFINE(node_id, init_fn, pm, data, config, level, prio, api)

EMUL_DT_DEFINE(node_id, init_fn, data, cfg, bus_api, backend_api)
```

- Parameters specific to emulators
  - **bus_api** - *bus messaging* (required) ✓
  - **backend_api** - *test scenario setup* (optional but useful) ✓

Creating an emulator - *bc12 example*



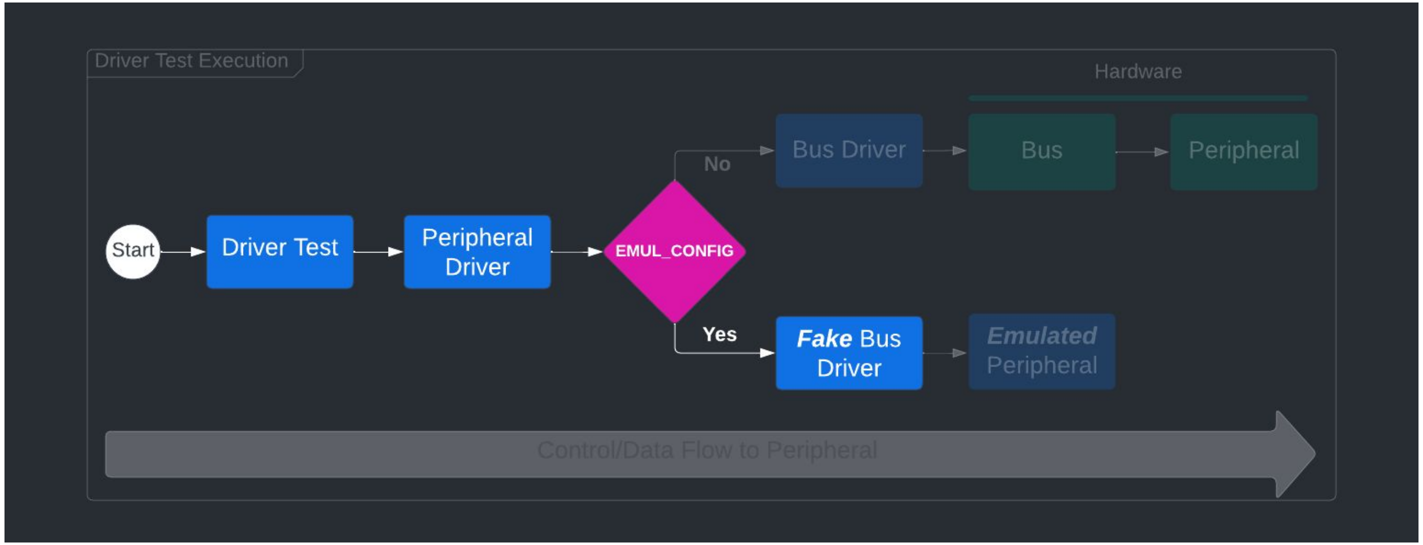
Agenda

- What is an emulator (peripheral emulator)?
- When and why should we emulate?
- Creating an emulator
- ***Using emulators***
- Q&A

Using an emulator - *What do we need?*

- Enable CONFIG_EMUL
- Device-Tree Nodes
  - Bus Emulator Node: intercepts bus driver to peripheral messages
  - Device Driver Node: builds driver + associated emulator

Using an emulator - Bus Emulator node



```
// Test invoking API
fuel_gauge_get_prop(...)
```

```
// Driver bus read
i2c_burst_read_dt(...)
```

```
// Fake Bus Controller
i2c_emul_transfer(...)
```

Using an emulator - I2C *Bus emulator node*

```
i2c0: i2c@100 {
        status = "okay";
        compatible = "zephyr,i2c-emul-controller";

        // Unused but included to conform as I2C controller
        clock-frequency = <I2C_BITRATE_STANDARD>;
        #address-cells = <1>;
        #size-cells = <0>;
        reg = <0x100 4>;
};
```

Using an emulator - *Device Node*

```
&i2c0 {
    max17048: max17048@36 {
        compatible = "maxim,max17048";
        reg = <0x36>;
        status = "okay";
    };
};
```

Examples

- Example emulators
  - BC12 Emulator & Backend API
  - AKM 09918c Emulator
- Example tests
  - BC12 Tests
  - Fuel Gauge Tests

Future Emulator Improvements - I2C *bus_api*

```
akm09918c_emul_transfer_i2c(const struct emul *target,
                            struct i2c_msg *msgs,
                            int num_msgs, int addr)
```

LOTS OF BOILER PLATE! :(

https://github.com/zephyrproject-rtos/zephyr/issues/59211

Future Emulator Improvements - More Emulators!

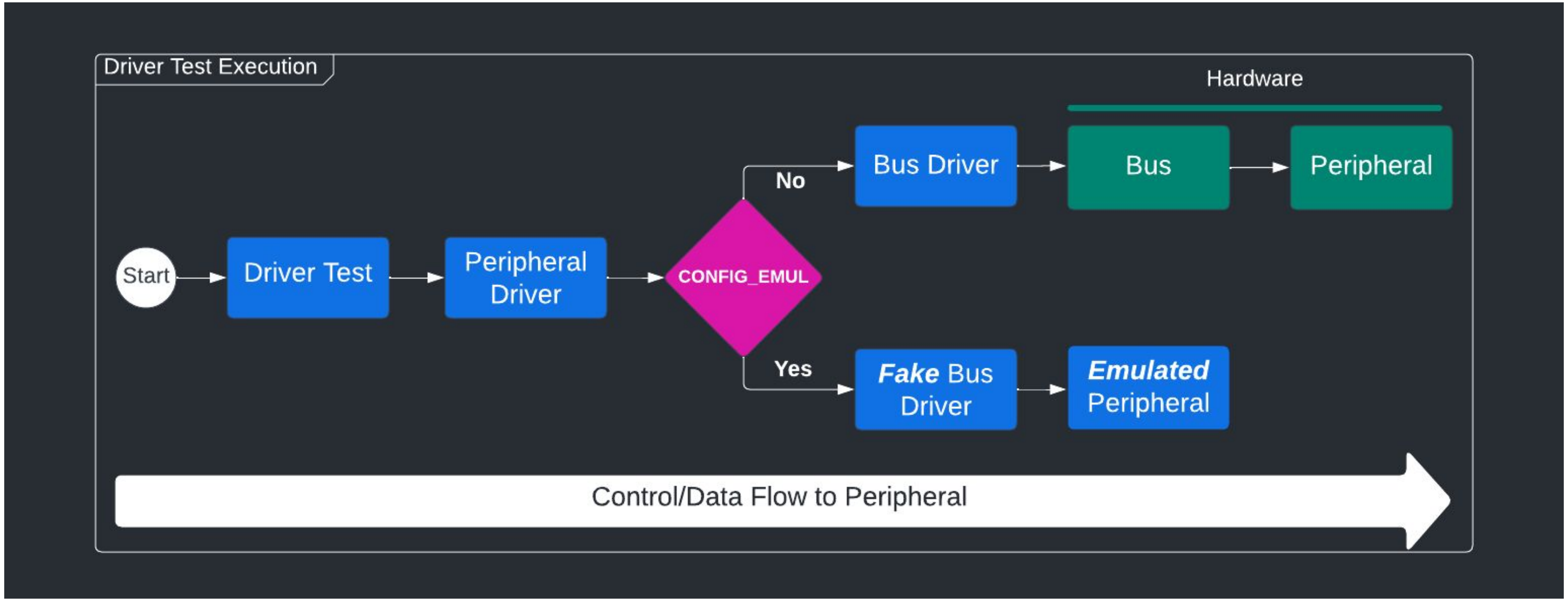
- Hundreds of Drivers in Zephyr with < 10 emulators
    - ADC - ads114s0x
    - GPIO - nct38xx
    - Sensor - bq274xx
- Ask new drivers to have emulator + tests

Success Stories at Google

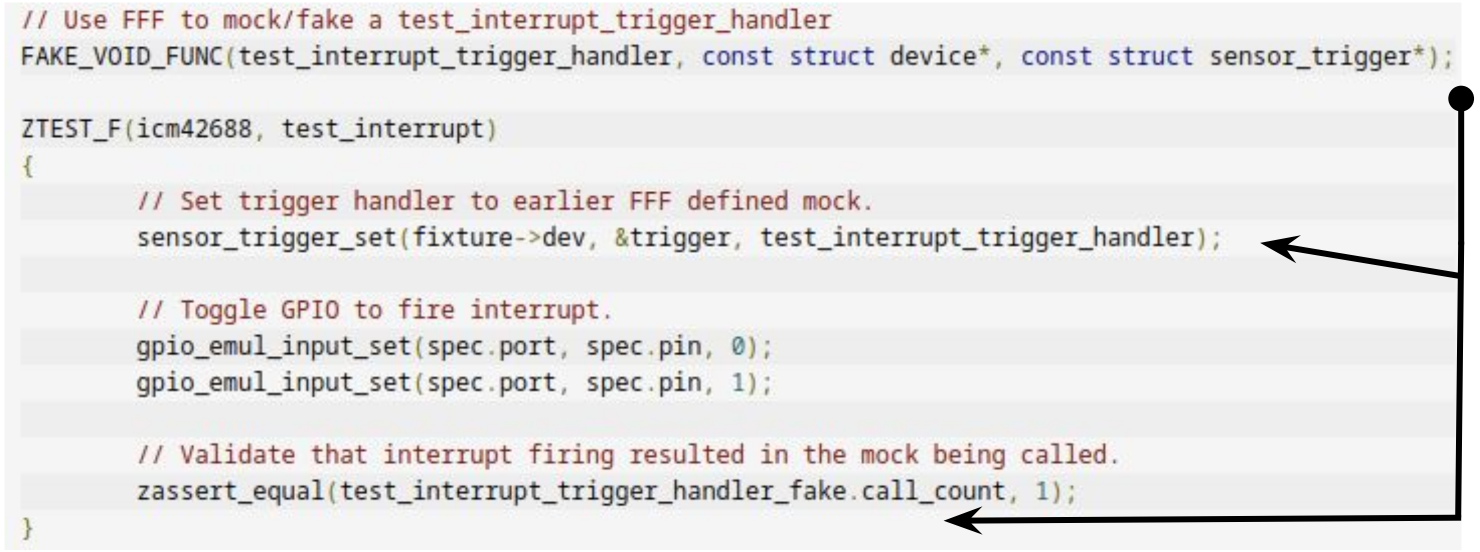
- **30%** ➡ **90%** code coverage
- Caught driver bugs
- Detected dead code
- Blocked regressions
- Rapidly prototyped drivers

Zephyr® Project
Developer Summit

Questions?



Mocks, Stubs, & Simple Fakes



```
// Use FFF to mock/fake a test_interrupt_trigger_handler
FAKE_VOID_FUNC(test_interrupt_trigger_handler, const struct device*, const struct sensor_trigger*);

ZTEST_F(icm42688, test_interrupt)
{
        // Set trigger handler to earlier FFF defined mock.
        sensor_trigger_set(fixture->dev, &trigger, test_interrupt_trigger_handler);

        // Toggle GPIO to fire interrupt.
        gpio_emul_input_set(spec.port, spec.pin, 0);
        gpio_emul_input_set(spec.port, spec.pin, 1);

        // Validate that interrupt firing resulted in the mock being called.
        zassert_equal(test_interrupt_trigger_handler_fake.call_count, 1);
}
```

Adapted from tests/drivers/sensor/icm42688/src/main.c#L223

Mocks, Stubs, & Simple Fakes	**Peripheral Emulators**
Function Model	Bus Transaction Model
Coupled to test	Coupled to driver/peripheral
Requires unit_testing or emulation	Enhanceable with mocks + etc.

Function stubs are not enough!

Using an emulator - ESPI *Bus emulator node*

ESPI included in native_posix.dts

```
espi0: espi@300 {
        status = "okay";
        compatible = "zephyr,espi-emul-controller";
        reg = <0x300 4>;
        #address-cells = <1>;
        #size-cells = <0>;
};
```

Using an emulator - SPI *Bus emulator node*

SPI included in native_posix.dts

```
spi0: spi@200 {
	status = "okay";
	compatible = "zephyr,spi-emul-controller";
	clock-frequency = <50000000>;
	#address-cells = <1>;
	#size-cells = <0>;
	reg = <0x200 4>;
};
```

Emulators / Qemu - Comparison

Emulator	**Qemu**
Simulate bus-peripheral interaction	Simulate entire Machine/Architecture
Based on a small Zephyr subsystem	Based on a large & solely separate project
Part of the Zephyr binary	Comprises Board defs. & Qemu configurations
Great for validating drivers	Great for validating architecture-specific code
Simple but **limited**	**Complete** but **complex**

We can use peripheral emulators in a Zephyr image built for Qemu!