Generating trustworthy hardware/software I^{2}C drivers for board management controllers

Zikai Liu Daniel Schwyn Prof. Timothy Roscoe

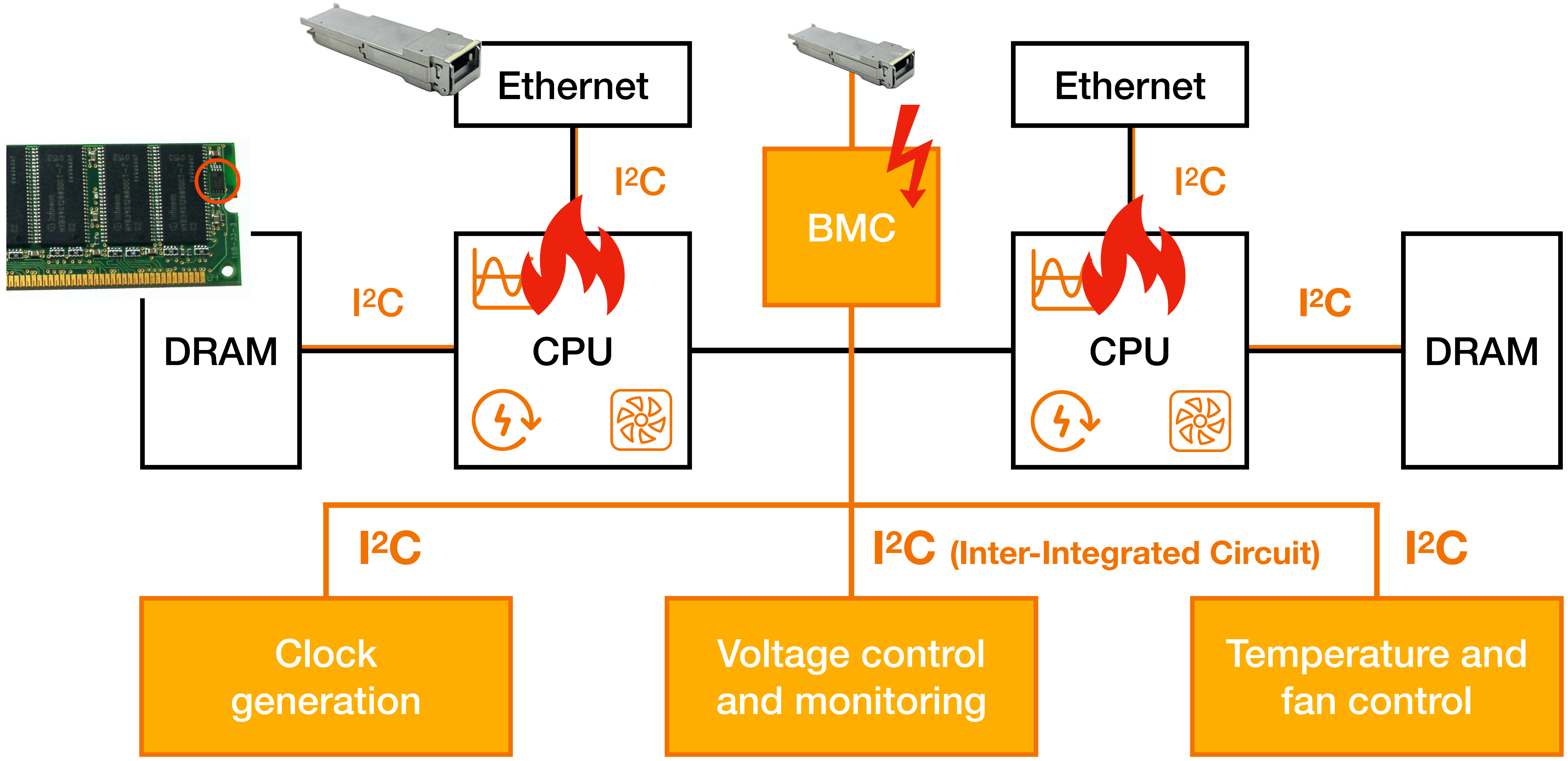
ETH zürich D INFK

seL4 Summit 2024, October 16

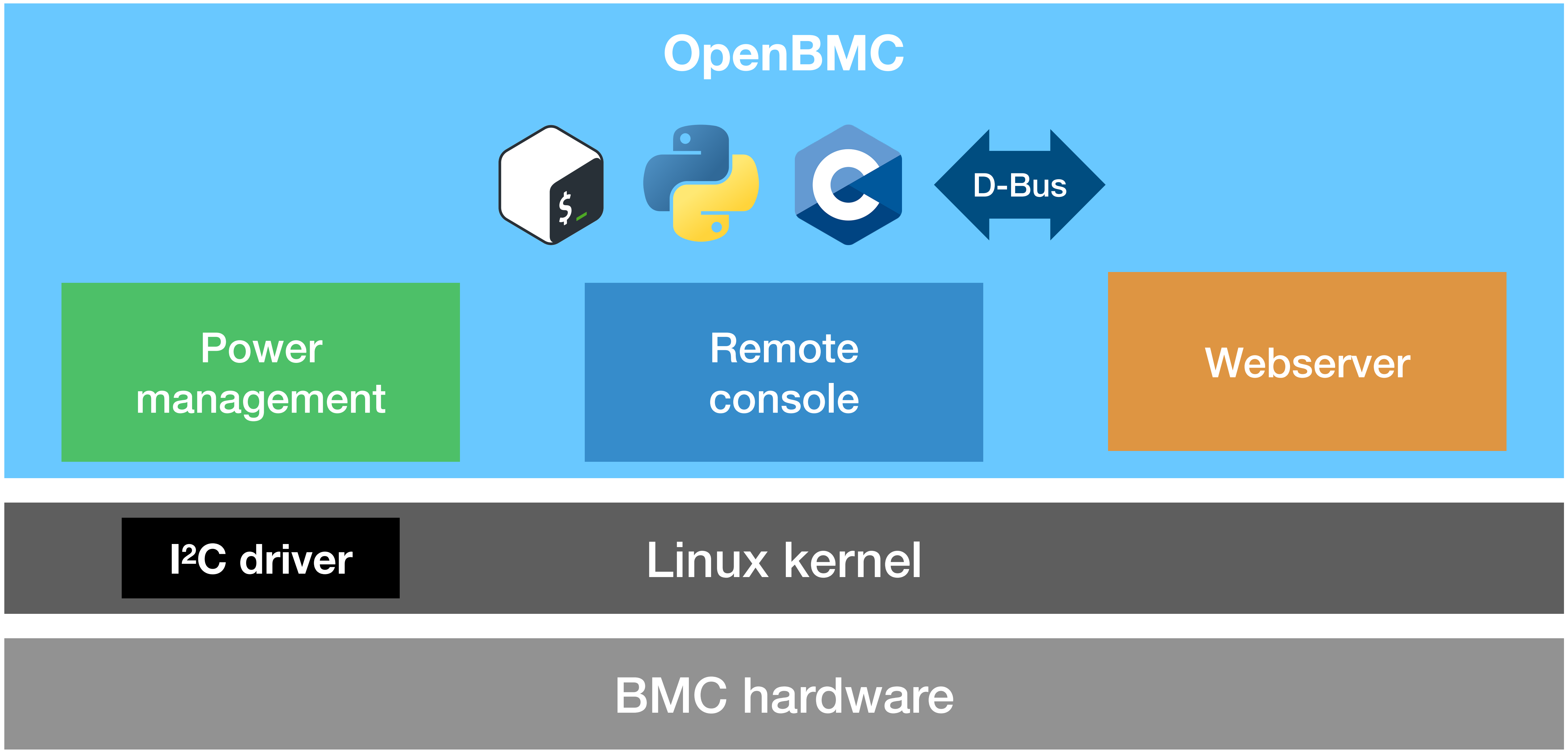
ENZIAN

Systems@ETH zürich

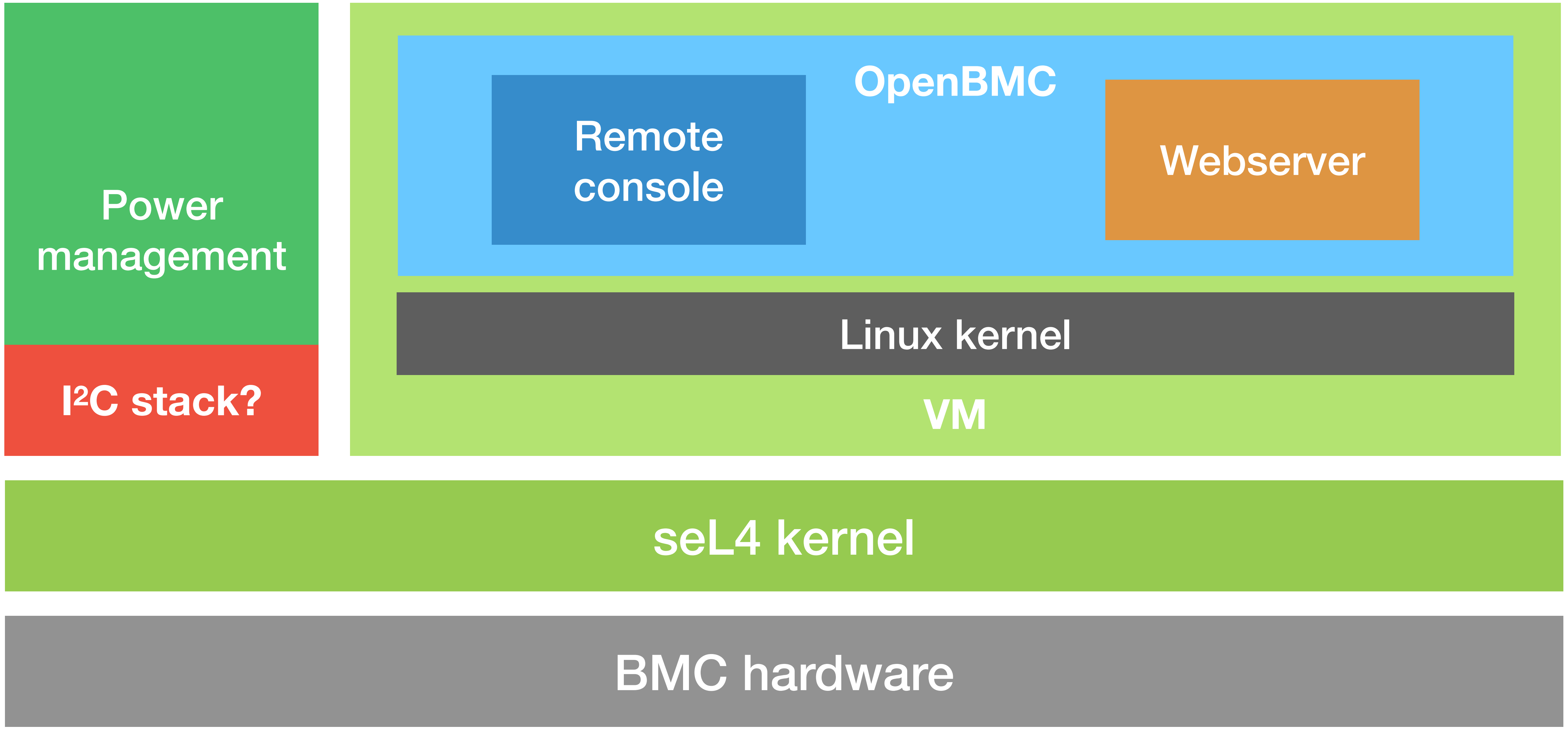
All behind the scene


Ethernet
Ethernet
BMC
I²C
I²C
I²C
I²C
DRAM
CPU
CPU
DRAM
I²C
I²C (Inter-Integrated Circuit)
I²C
Clock generation
Voltage control and monitoring
Temperature and fan control

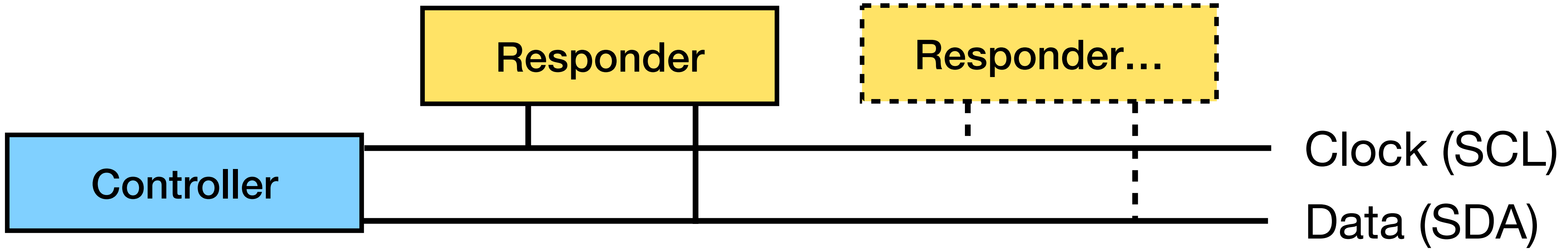
State-of-the-art BMC firmware is not trustworthy



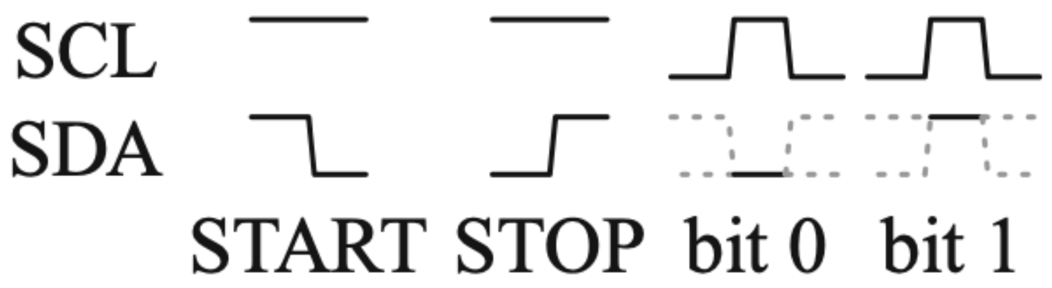
Solution: secure BMC with seL4 via cyber retrofit



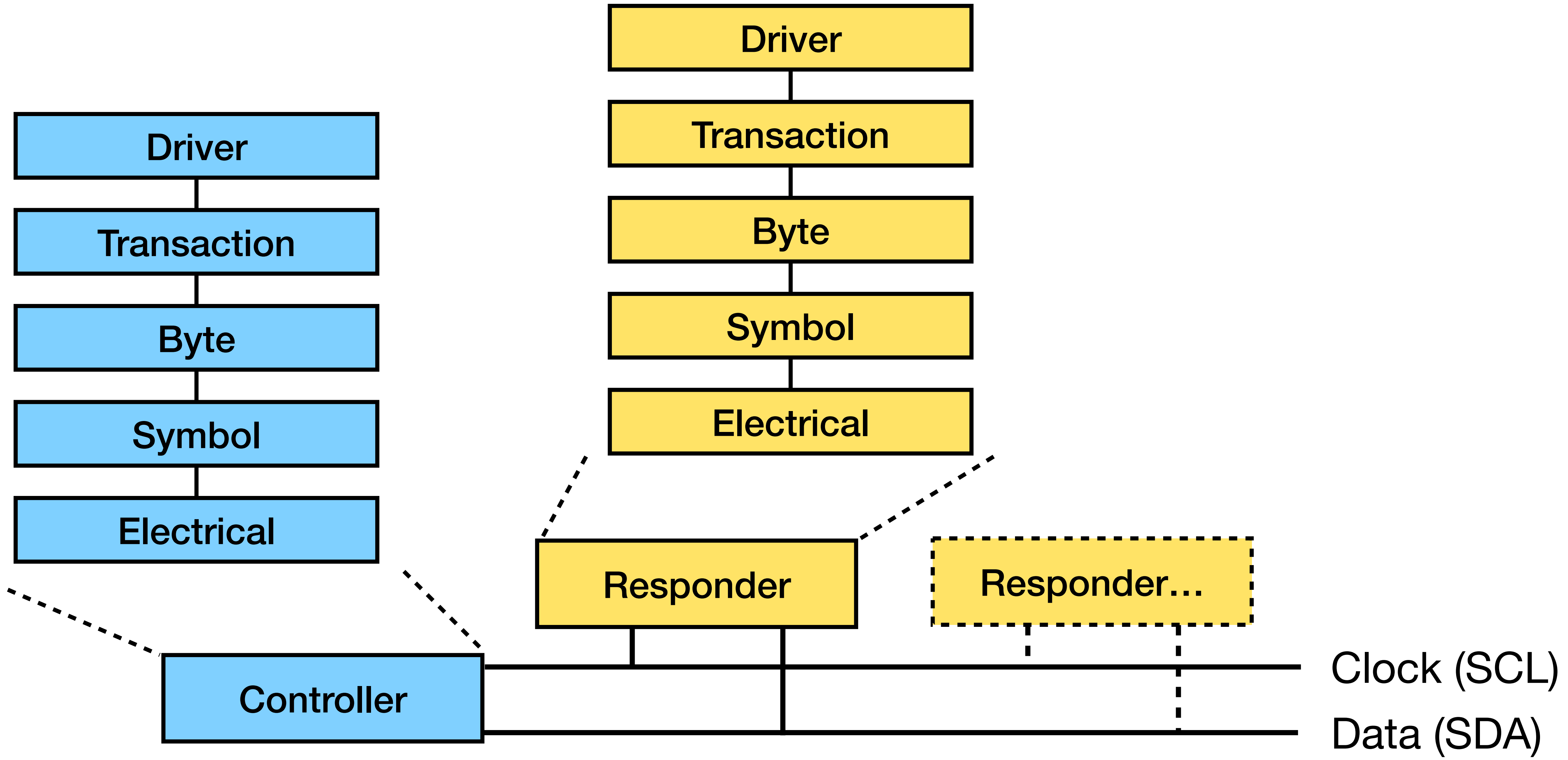
I²C is a bus-based protocol



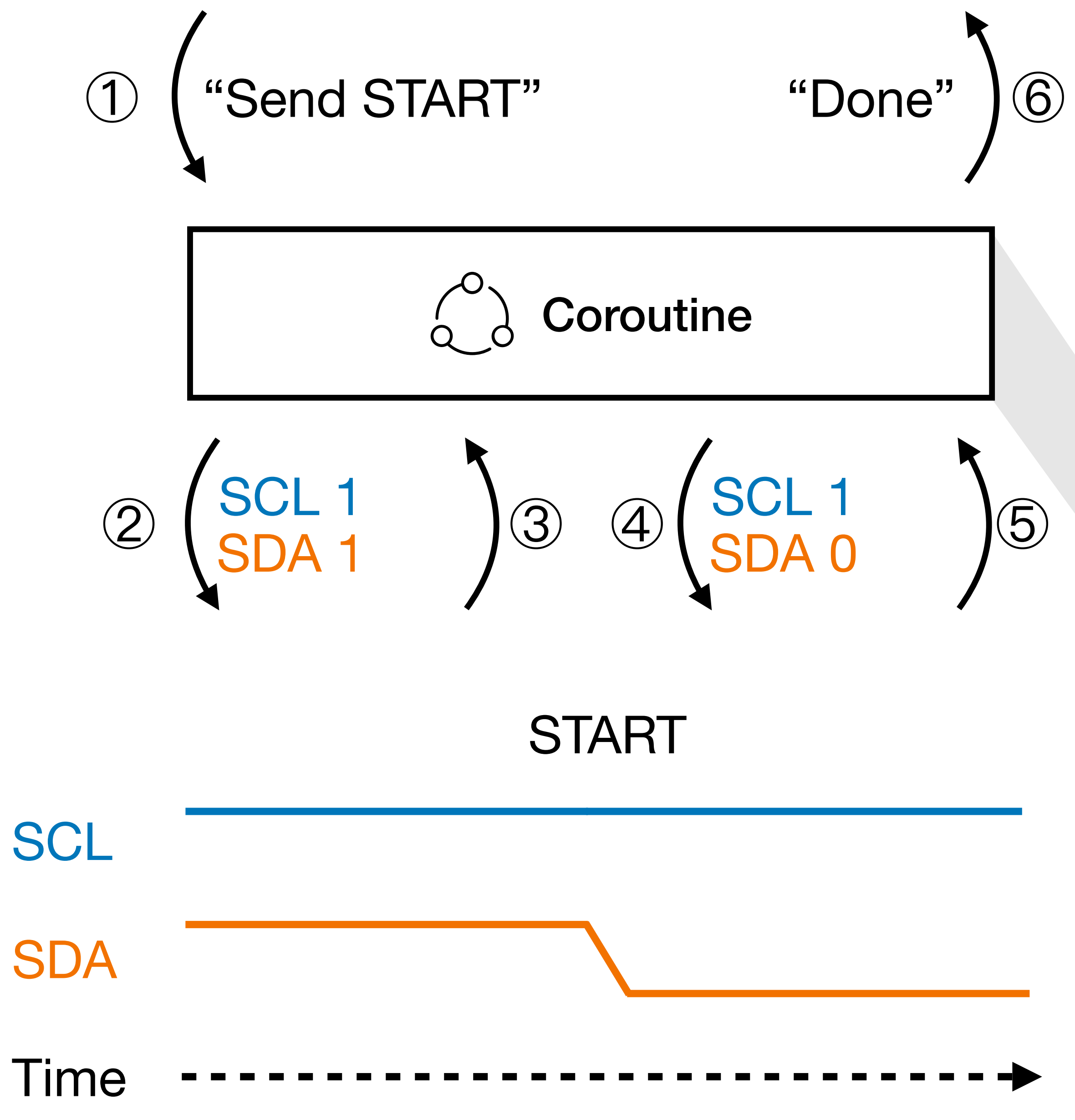
4 symbols

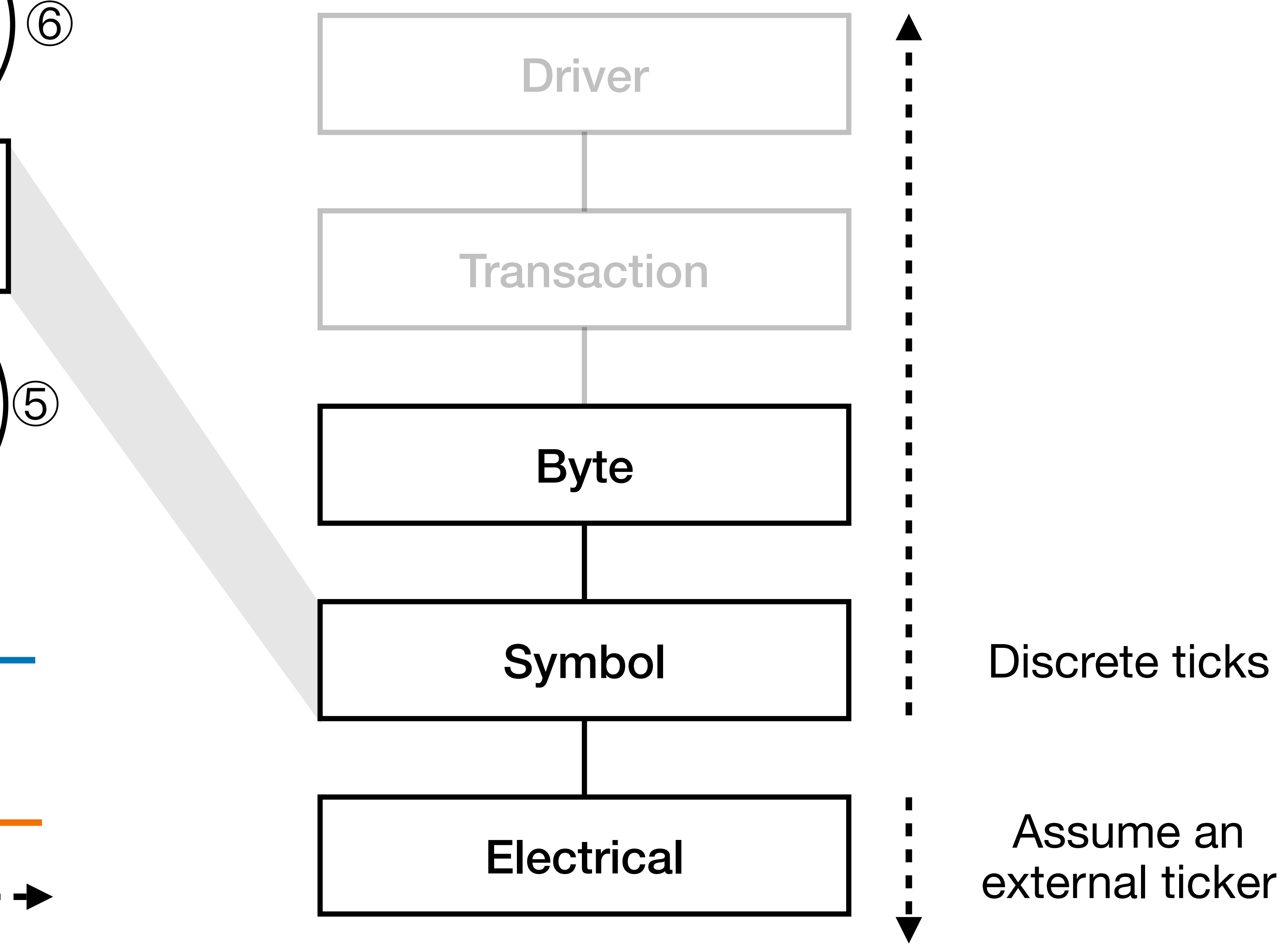


I²C is a layered protocol

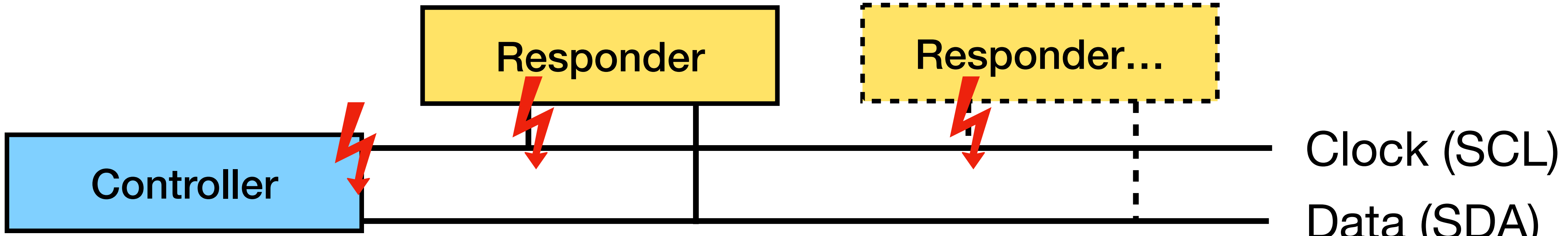


I²C is a layered protocol

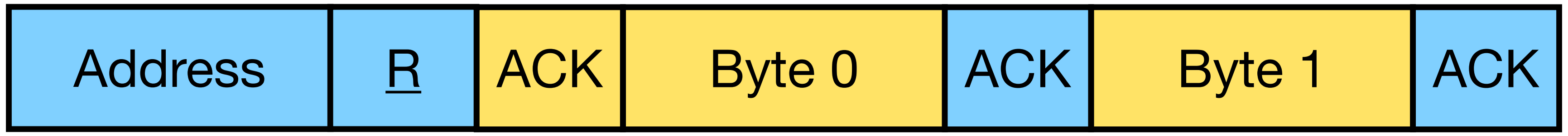




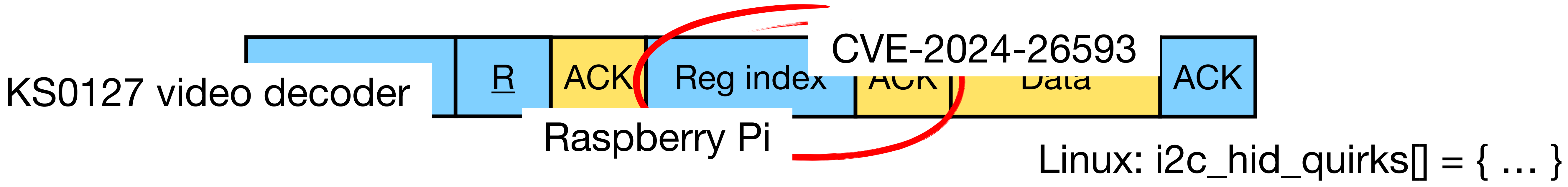
Only if everything conforms to the standard…



I²C standard <u>read</u> (transaction)

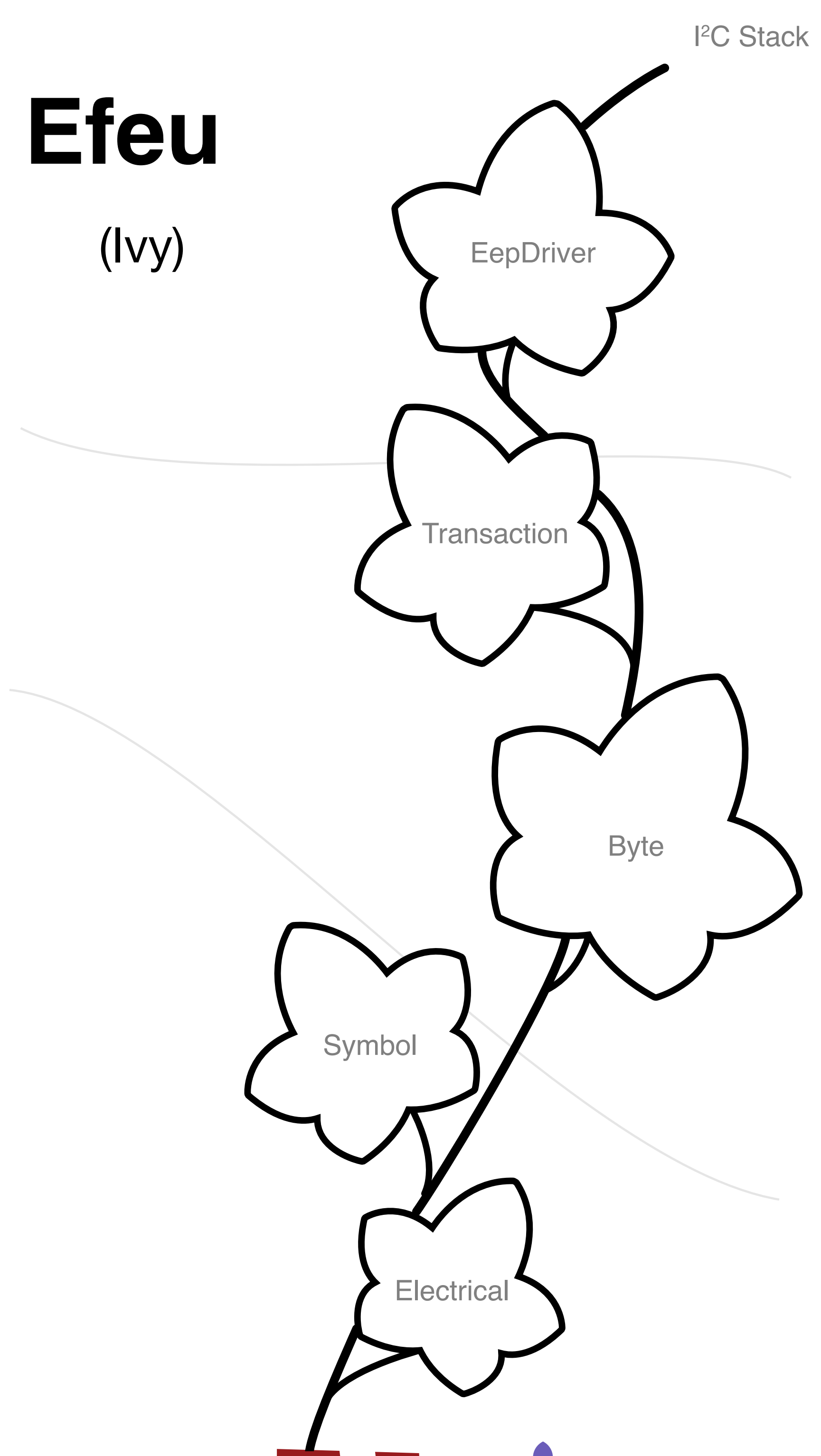


AS5011 hall sensor <u>read</u> (its register)



Efeu

(Ivy)



$$\begin{tikzcd}
 & \text{Promela} \arrow[r] & \text{SPIN model checker} \arrow[dl, dashed, bend left] \arrow[ddl, dashed, bend left] \\
\text{Spec} \arrow[ur] \arrow[r] \arrow[dr] & \text{C} \arrow[r] & \text{seL4} \\
 & \text{HDL} \arrow[r] & \text{FPGA}
\end{tikzcd}$$

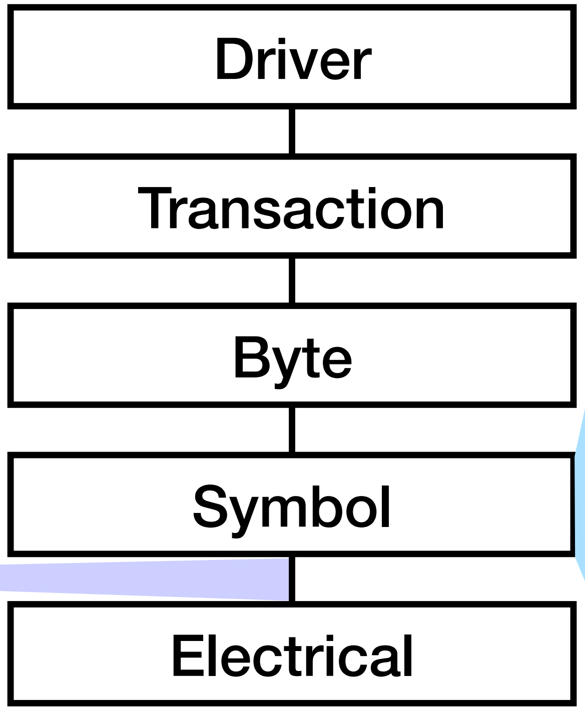
Efeu: generating efficient, verified, hybrid hardware/software drivers for I2C devices
EuroSys'25 (to appear)

Specify the whole system

ESI: Efeu System Information

```
layer Electrical;
layer Symbol;
interface <Electrical, Symbol> {
  => {
    bit scl;
    bit sda;
  },
  <= {
    bit scl;
    bit sda;
  },
};
```

A light-weight DSL



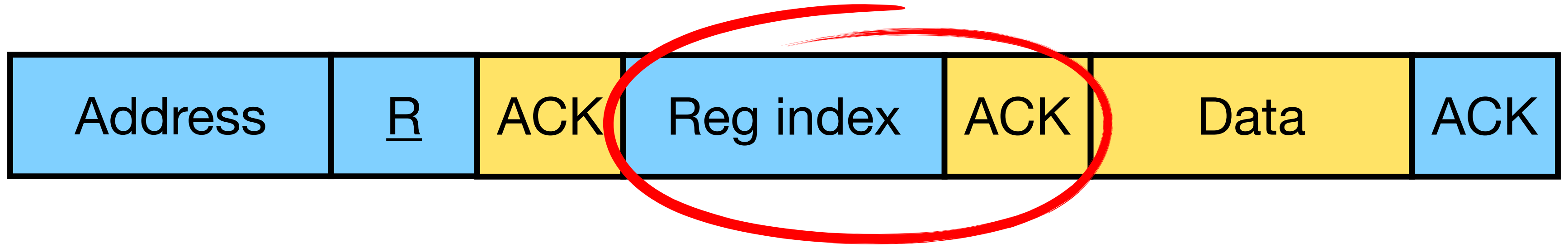
ESM: Efeu State Machine

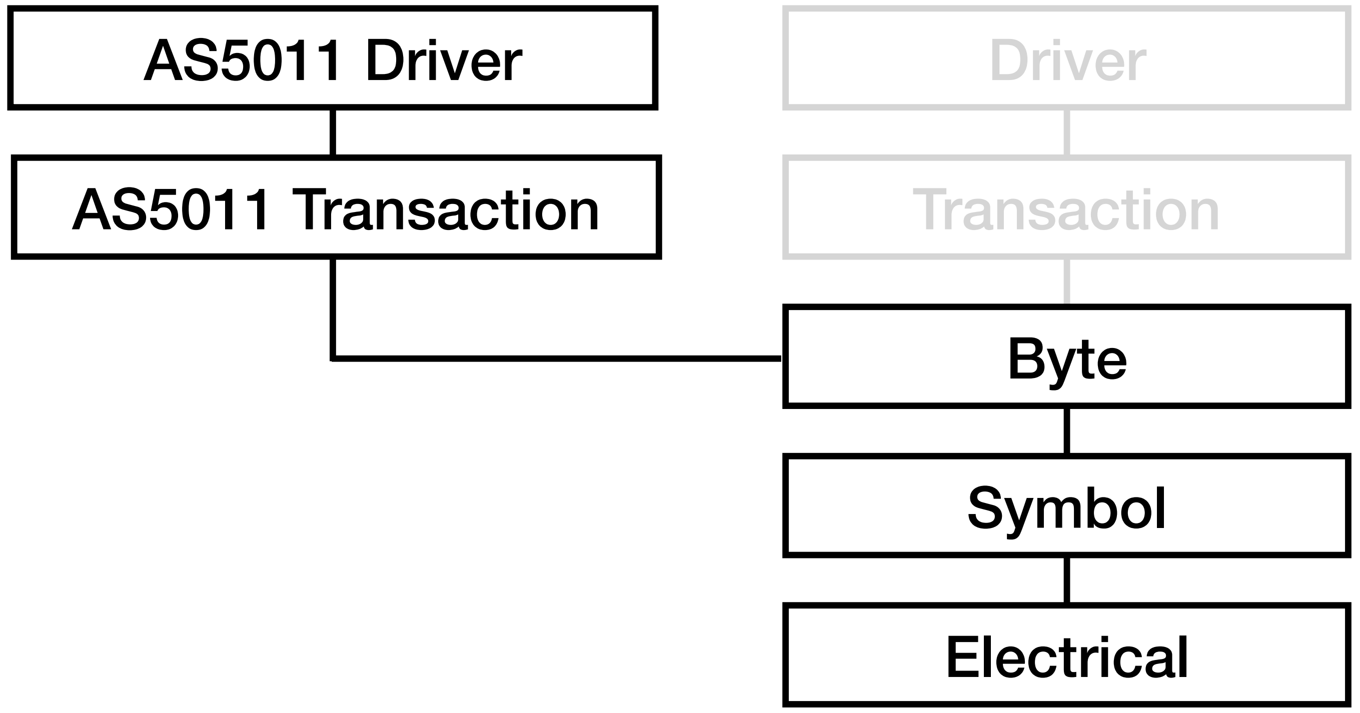
```
void Symbol() {
  ByteToSymbol b;
  ElectricalToSymbol e;
  ...
NEXT:
  b = talkByte(DONE);
  if (b.symbol == START) {
     e = talkElectrical(1, 1);
     e = talkElectrical(1, 0);
     goto NEXT;
  }
  ...
}
```

A subset-C DSL

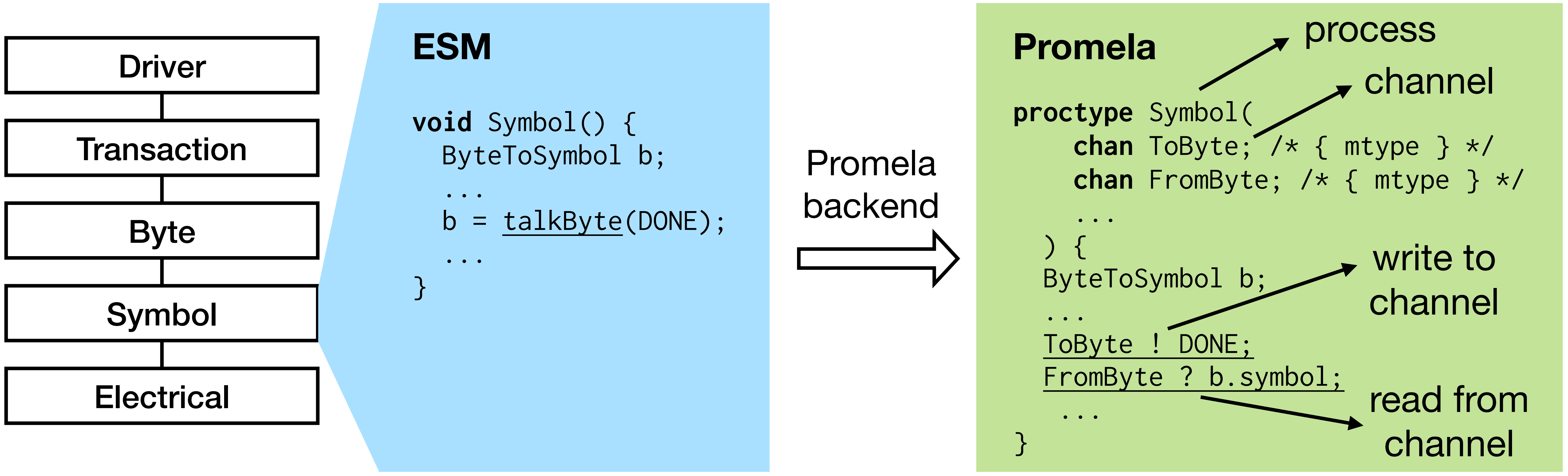
Reuse layers for non-compliant devices

AS5011 hall sensor read (its register)





Layers in Promela: processes



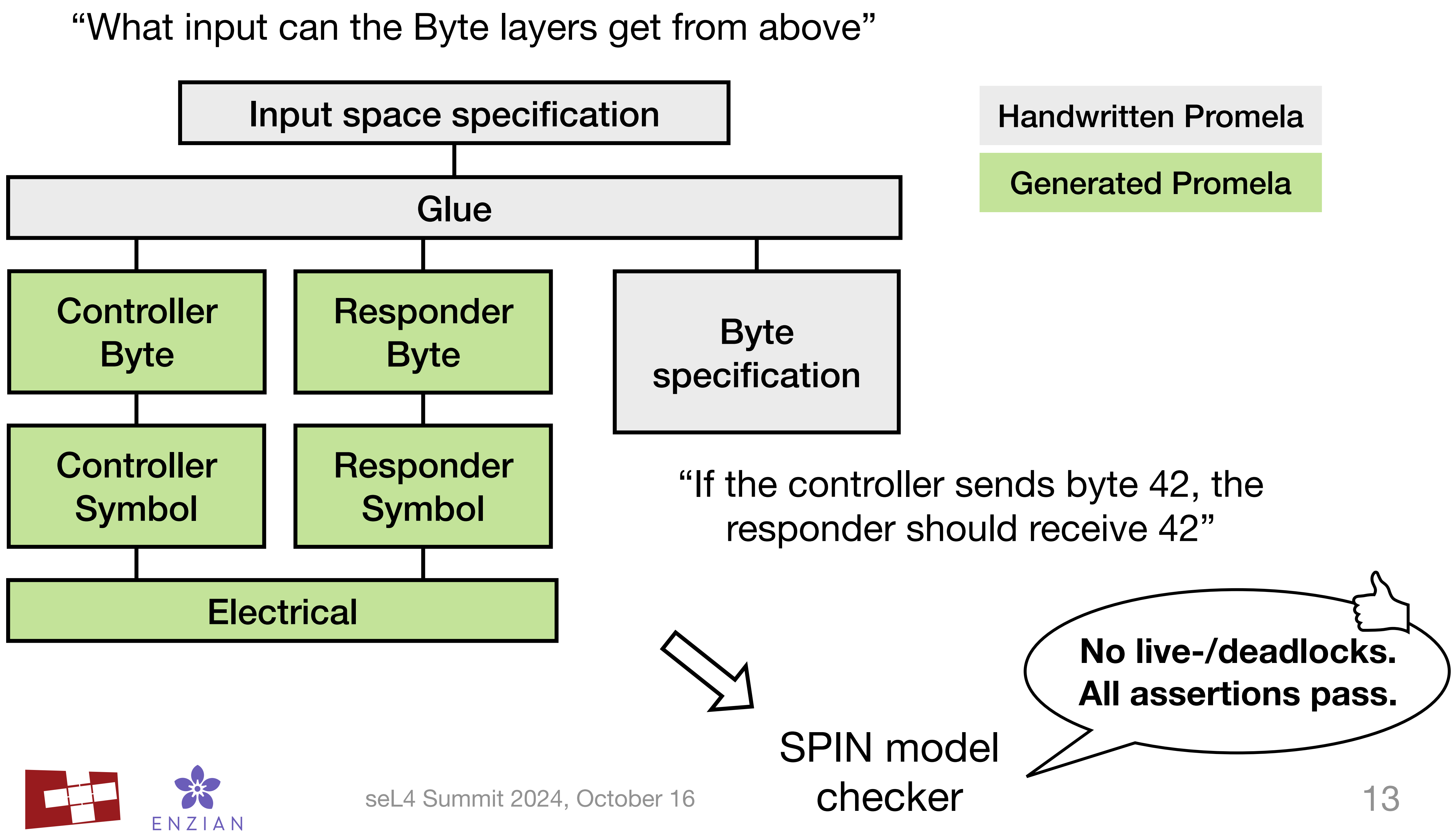
```
void Symbol() {
  ByteToSymbol b;
  ...
  b = talkByte(DONE);
  ...
}
```

```
proctype Symbol(
    chan ToByte; /* { mtype } */
    chan FromByte; /* { mtype } */
    ...
  ) {
  ByteToSymbol b;
  ...
  ToByte ! DONE;
  FromByte ? b.symbol;
   ...
}
```

Written

Generated

Verifier architecture


"What input can the Byte layers get from above"
Input space specification
Glue
Controller Byte
Responder Byte
Byte specification
Controller Symbol
Responder Symbol
Electrical
Handwritten Promela
Generated Promela
"If the controller sends byte 42, the responder should receive 42"
SPIN model checker
No live-/deadlocks. All assertions pass.

Unit under test (processes)

Verifiers and Assumptions

The assemblage of layer ____ and below conforms to the behavior specification

- 1 controller 1 responder:
  Symbol/Byte/Transaction/EEPROM driver
- 1 controller N responders:
  EEPROM driver
- KS0127 video encoder:
  Byte/Transaction
- Raspberry Pi:
  Byte w/ and w/o clock stretching

- Efeu compiler is trusted
- Downstream toolchains (compilers, EDA tools) are trusted
- Electrical layer correctly retimes the discrete ticks to the I^2C bus ticks

Promela backend: layer → process

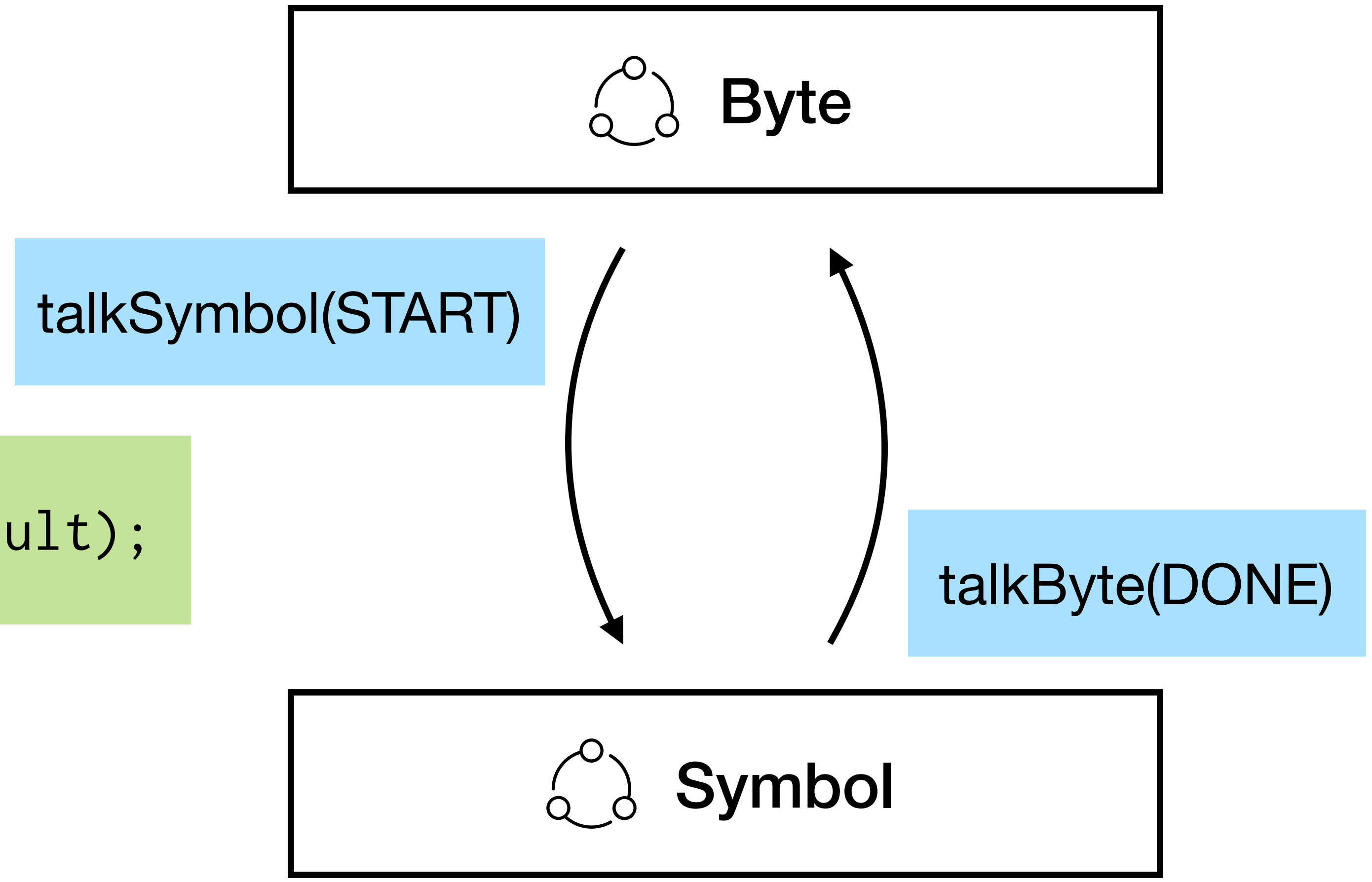
	ESM	Generated
		Promela
Protocol layer	Coroutine	<u>Process</u>
Inter-layer communication	`talk` derivative	<u>channel operations</u>

Lightweight software implementation

ESM

Generated C

```
void Byte(...) {...}
```



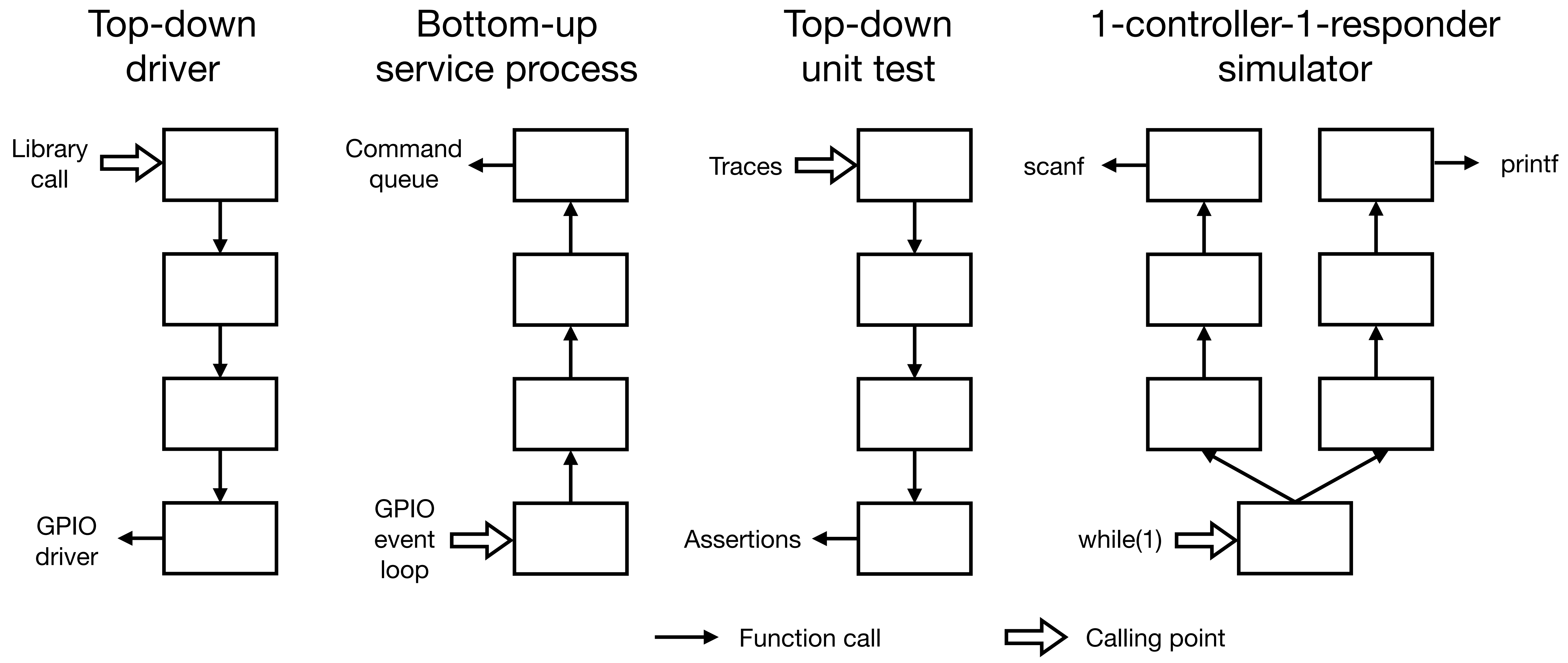
```
Symbol(START, &result);
```

```
void Symbol(...) {...}
```

```
switch(pos) {
  case 1: goto label1;
  ...
}


  *result = DONE;
  pos = 1;
  return;
label1:
```

Lightweight software implementation



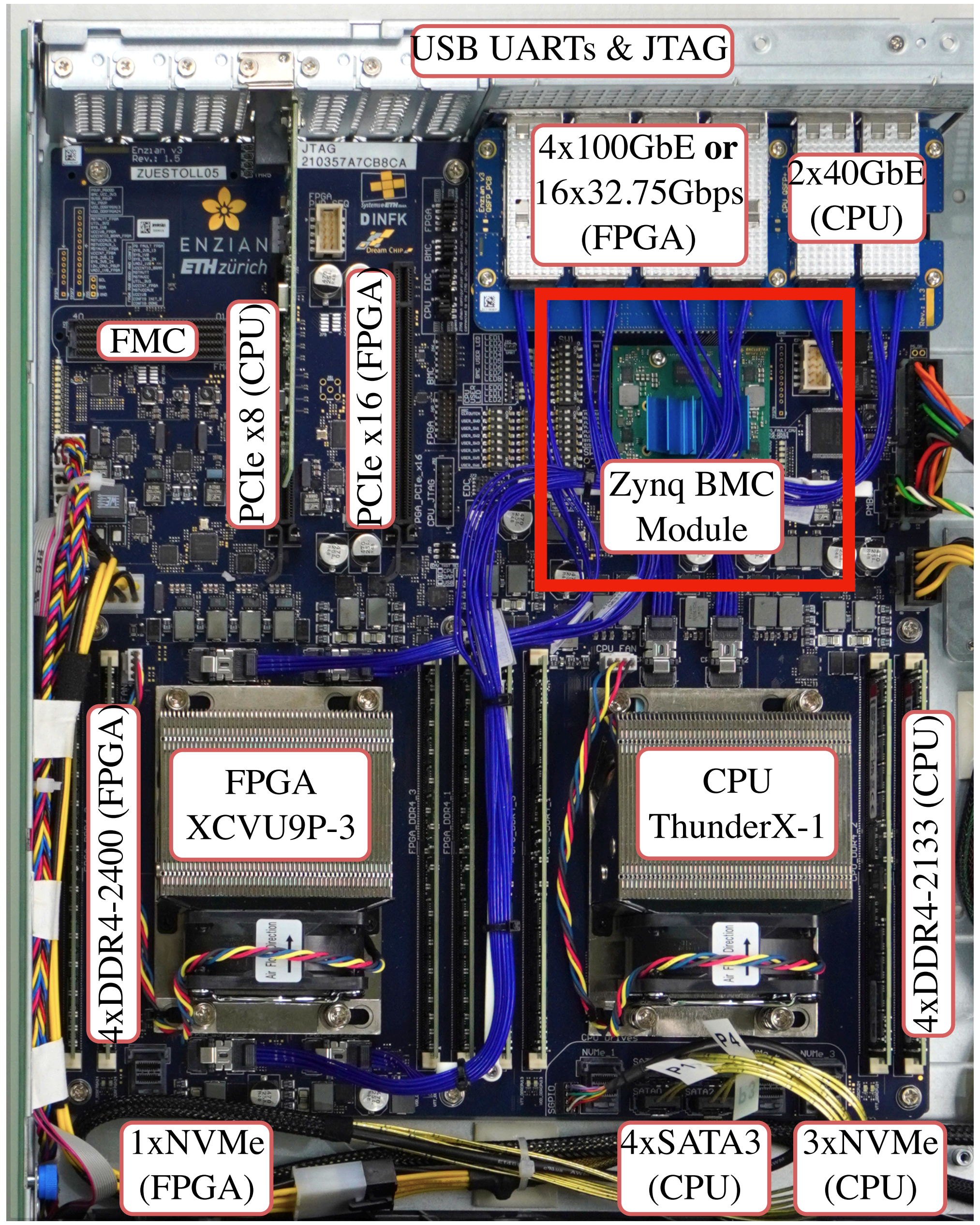
C backend: layer → coroutines

	ESM	Generated	
		Promela	C
Protocol layer	Coroutine	Process	Case statement-based coroutine
Inter-layer communication	`talk` derivative	channel operations	Case statement-based coroutine switch

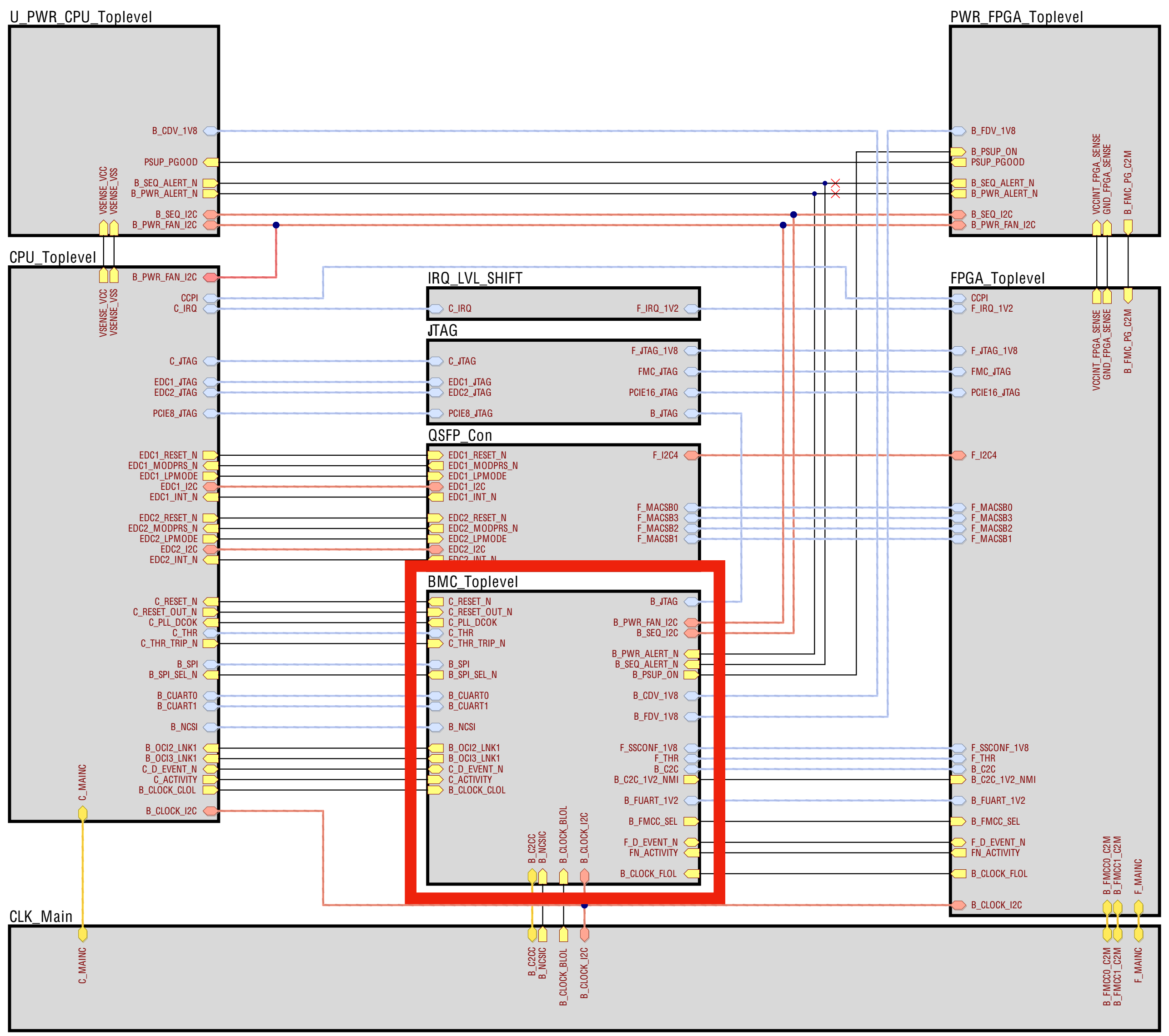
HDL backend: layer → Verilog module

	ESM	Generated		
		Promela	**C**	**HDL**
Protocol layer	Coroutine	Process	Case statement-based coroutine	Verilog module
Inter-layer communication	`talk` derivative	channel operations	Case statement-based coroutine switch	Handshaking protocol

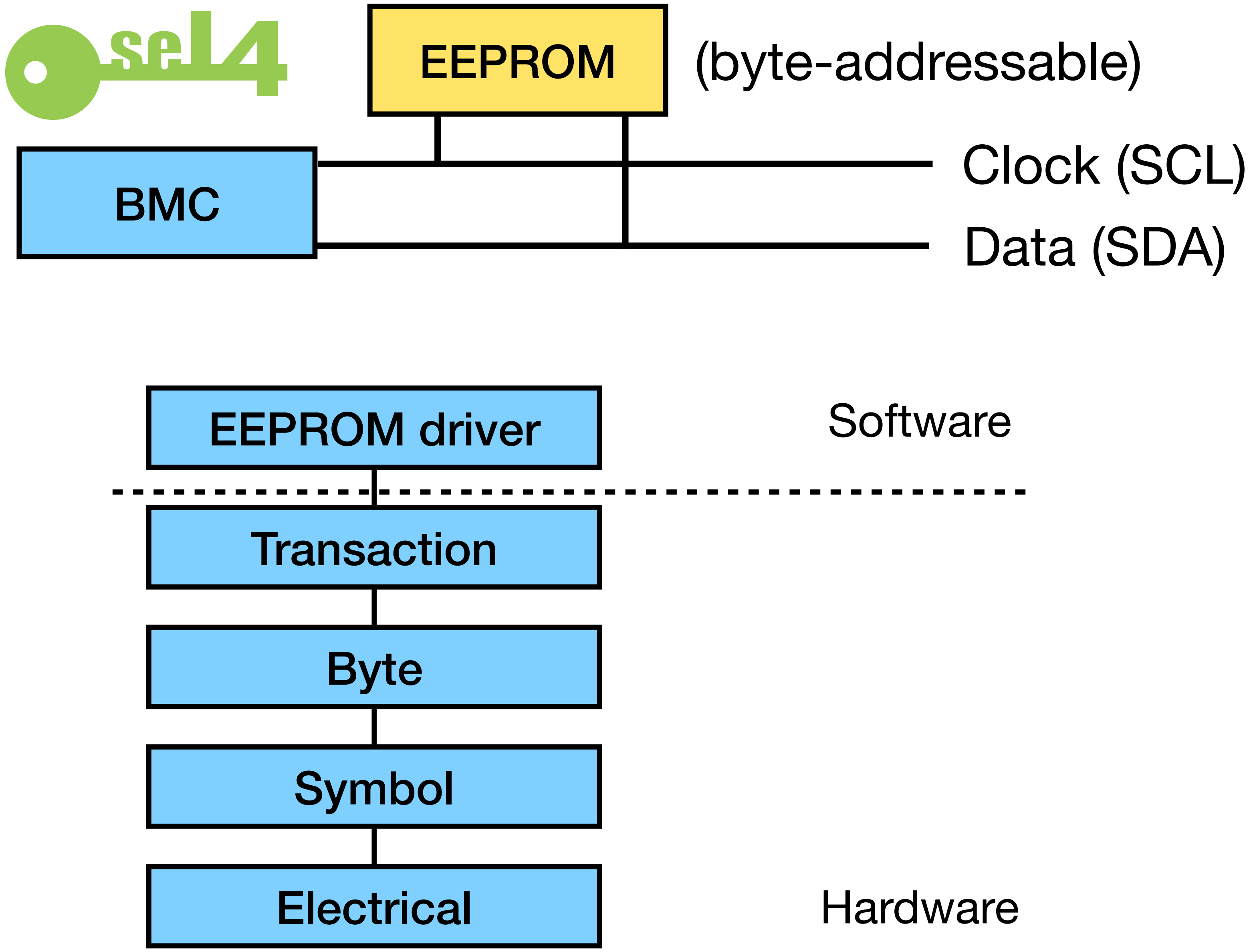
We know this because we build hardware



https://enzian.systems/



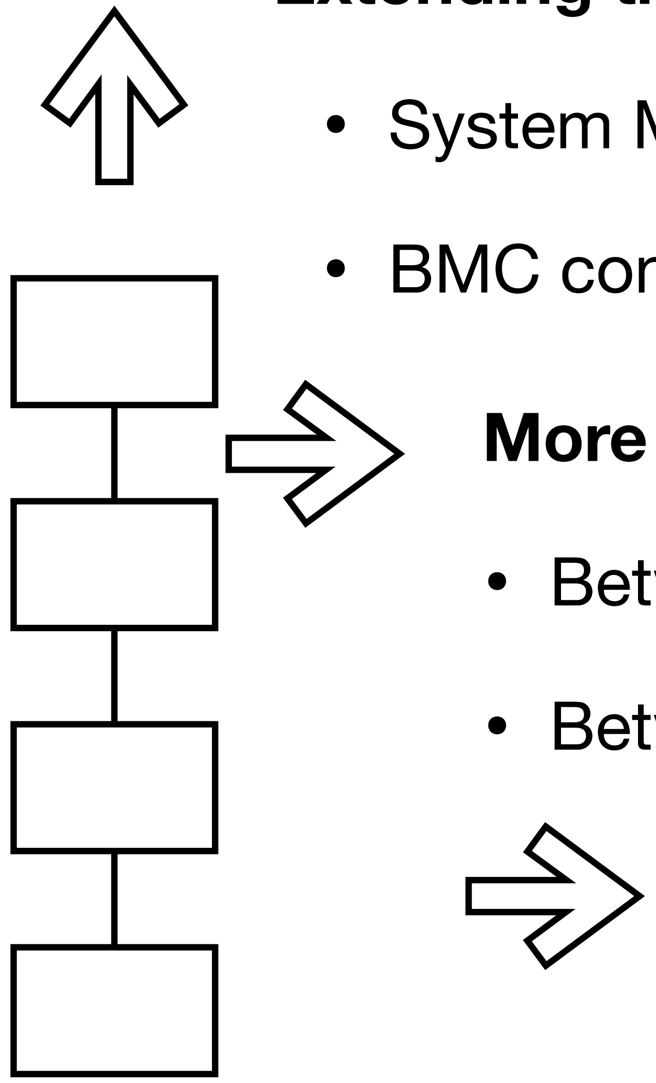
Demo!



What you will see:

- seL4 Microkit .system file
- MMIO helper functions using seL4 system calls
- Generated C code for the EEPROM driver layer
- Read/write the EEPROM

A glimpse into the future



Extending the stack vertically

- System Management Bus (SMBus)
- BMC control logic

More ways to “talk” between layers

- Between processes
- Between VMs and the host OS

Build stacks for other protocols

- Serial Peripheral Interface (SPI)
- Controller Area Network (CAN)

Efeu: generating efficient, verified, hybrid hardware/software drivers for I2C devices

EuroSys’25 (to appear)

zikailiu.com/about/efeu.pdf

Efeu compiler, models, FPGA designs… **All open source!**

https://gitlab.inf.ethz.ch/project-opensockeye/efeu

$$\begin{tikzcd}
 & \text{Promela} \arrow[r] & \text{SPIN model checker} \\
\text{Spec} \arrow[ur] \arrow[r] \arrow[dr] & \text{C} \arrow[r] & \text{Linux/seL4} \\
 & \text{HDL} \arrow[r] & \text{FPGA}
\end{tikzcd}$$

Talk to us!