Hardware/Software Co-Design with the Open Source Renode Framework and RISC-V

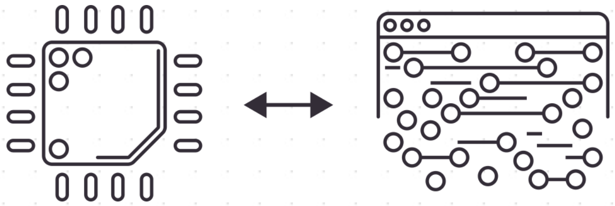
Getting Started With RISC-V NA Tour, 2019

Michael Gielda, mgielda@antmicro.com

antmicro

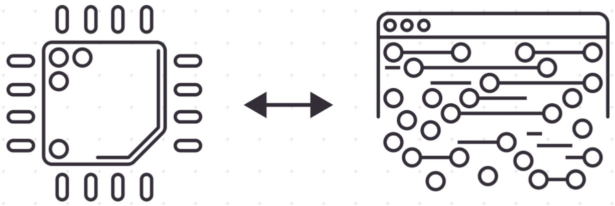
Why HW/SW co-design?

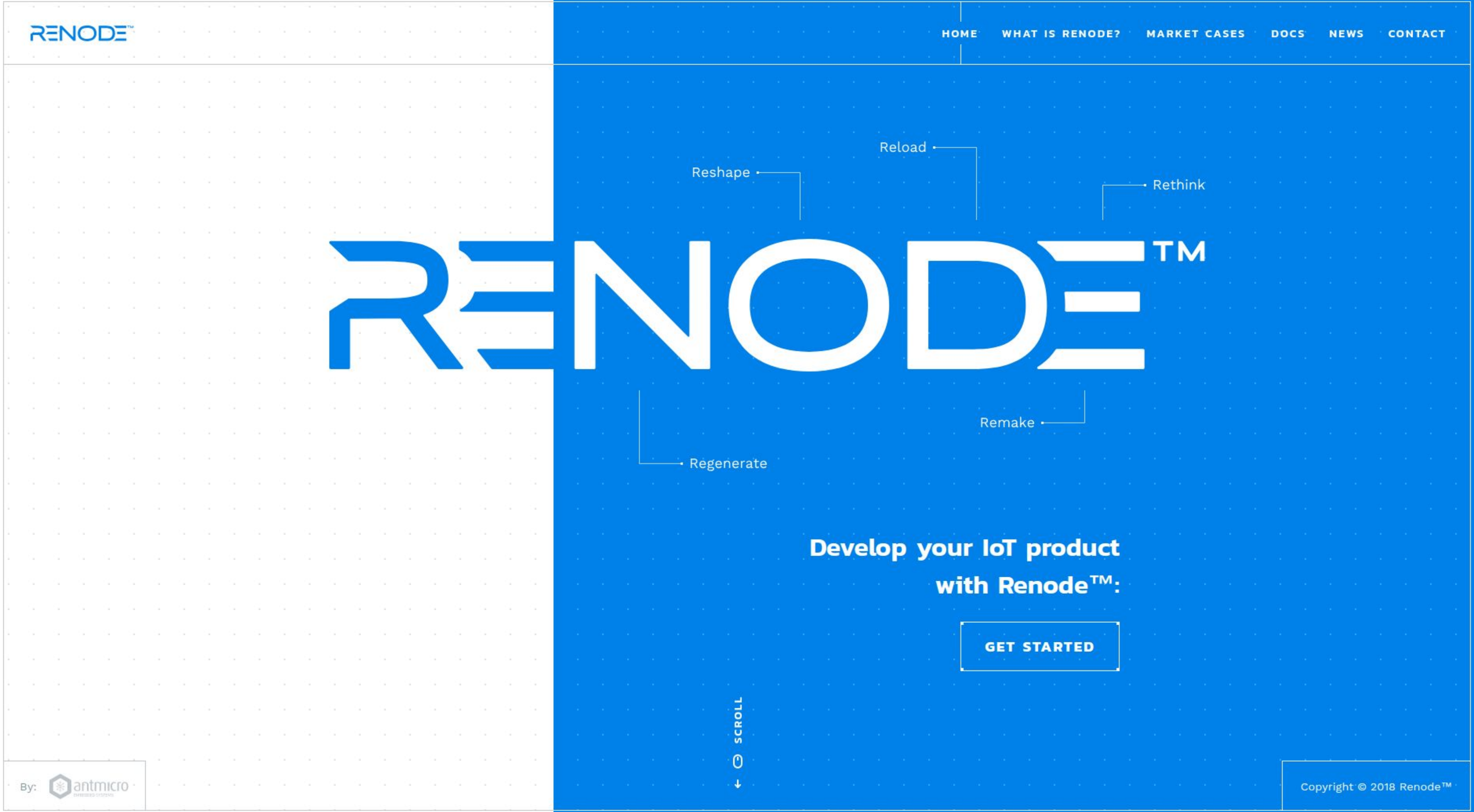
- Design cycles are radically decreasing, and new HW platforms become more complex: features vs security vs safety vs ease-of-use
- More and more software is expected to be provided for out of the box experience
- Configurability and openness of RISC-V offers the promise of "software-driven" design...



Why HW/SW co-design?

- ... but new tools and methods are needed to deliver on the promise
- You can’t just build your platform sequentially any more, many elements need to be set in motion simultaneously and iterated on
- Cost of bug caught early is 1000x less than one caught further down the line





RENODE™

HOME WHAT IS RENODE? MARKET CASES DOCS NEWS CONTACT

Reload

Reshape

Rethink

RENODE™

Remake

Regenerate

Develop your IoT product with Renode™:

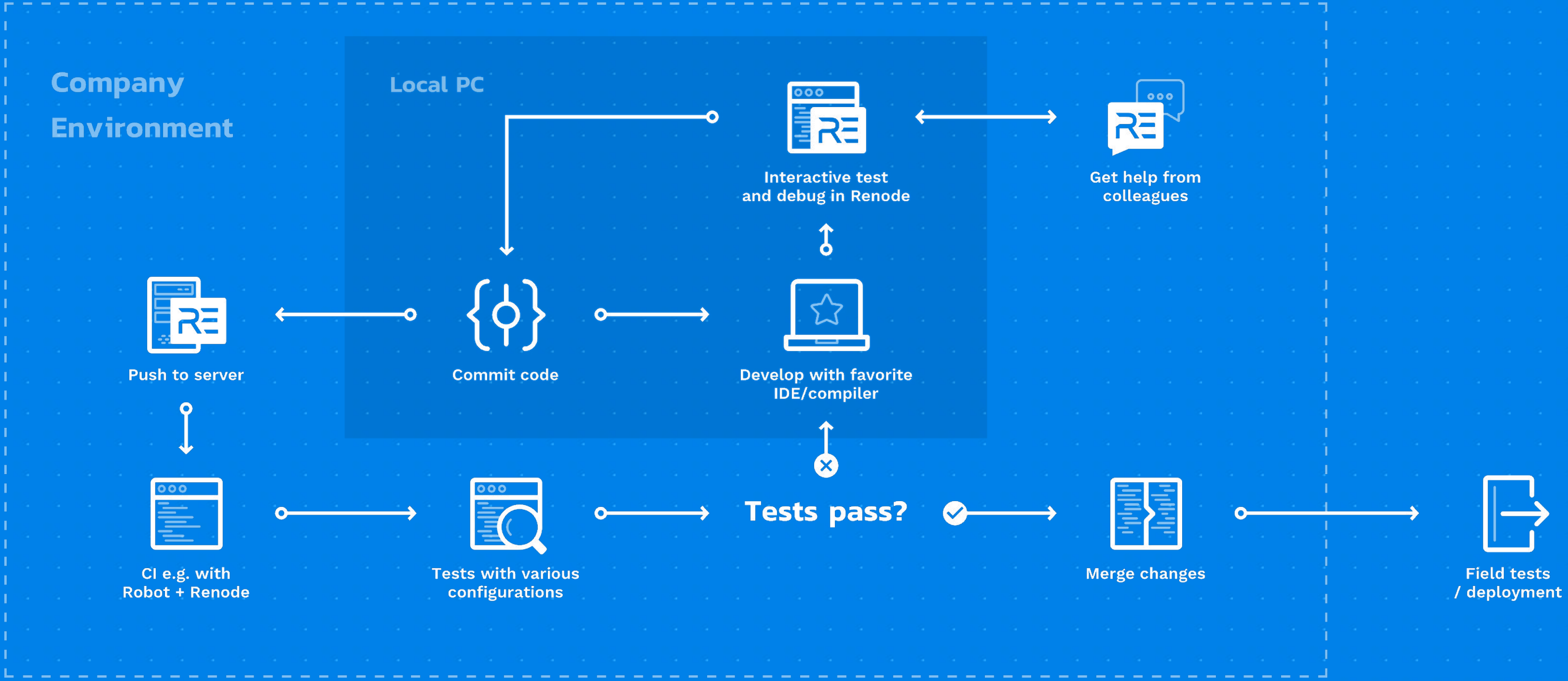
GET STARTED

SCROLL

By: antmicro

RENODE™

Continuous Integration based development


Company Environment
Local PC
Interactive test and debug in Renode
Get help from colleagues
Push to server
Commit code
Develop with favorite IDE/compiler
CI e.g. with Robot + Renode
Tests with various configurations
Tests pass?
Merge changes
Field tests / deployment

Renode in short

- rapid development framework for building software including real, end-user applications
- open source Instruction Set Simulator (ISS) with a multi-layered framework on top
- strong background in practical use across multiple architectures (mainly but not only ARM & RISC-V)
- commercial backing from Antmicro - we use it ourselves extensively

Renode – why?

- system emulator - mimic entire boards or even multiple connected nodes
- most interesting name I heard: “hierarchical functional simulator” - ‘building-block’ nature
- scriptable, API-oriented, extremely flexible
- software agnostic - runs Zephyr, Linux, bare metal, proprietary SW: binary compatible (no special compilation targets)

RENODE™ &

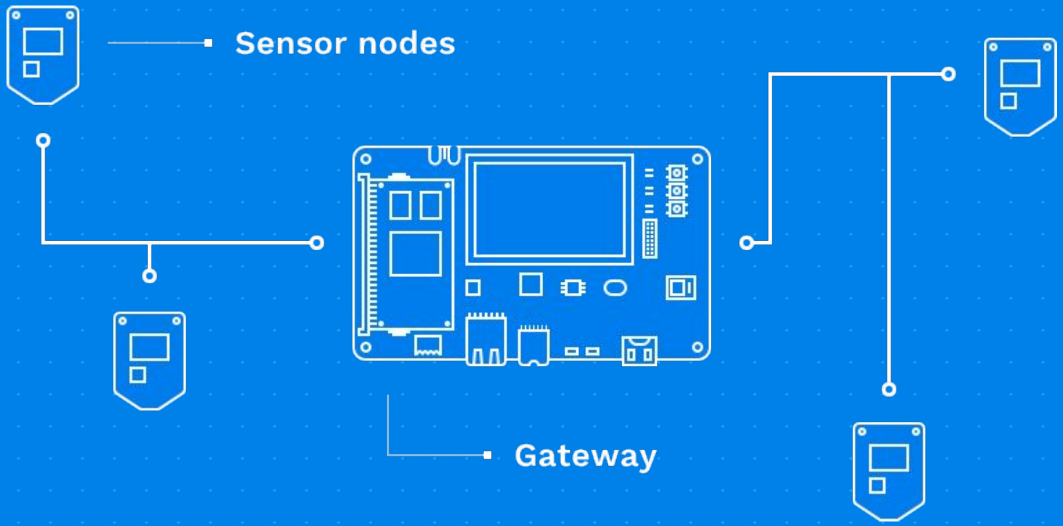
RISC-V

Easy to develop & prototype

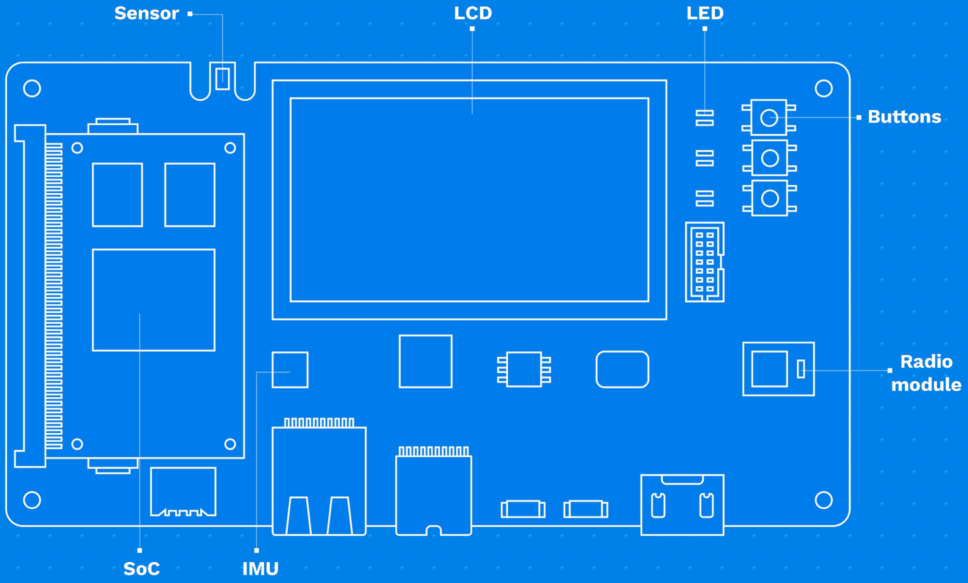
- lots of useful abstraction and interfaces
- human readable, modular and extensible platform description format
- plug-and-play blocks, Python stubs

```
uart: UART.MiV_CoreUART @ sysbus 0x70001000
    clockFrequency: 66000000

cpu: CPU.RiscV @ sysbus
    cpuType: "rv32g"

plic: Interrupts.PlatformLevelInterruptController @ sysbus 0x40000000
    IRQ -> cpu@1
    numberOfSources: 31 //based on release notes
```

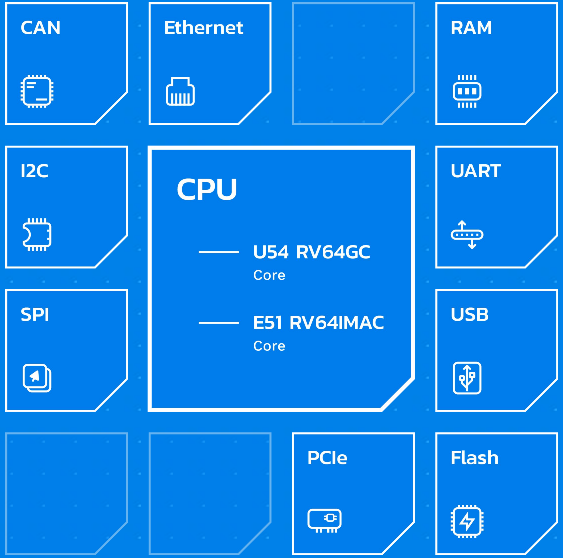
Layer #3: Complex system



Layer #2: The device



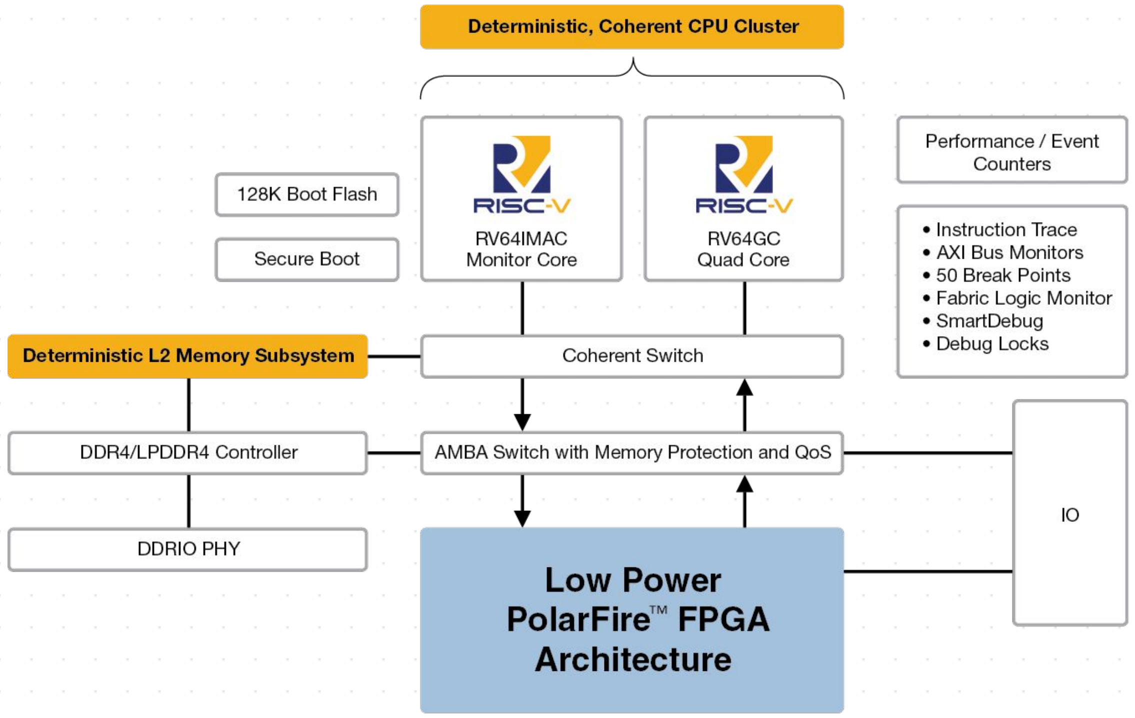
Layer #1: System-on-Chip



Enabling the Freedom to Innovate: PolarFire SoC FPGA architecture

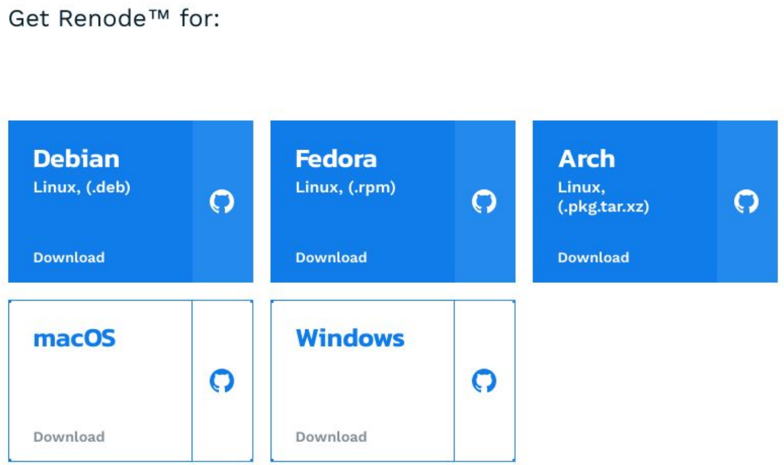
Microsemi® a MICROCHIP company

PolarFire SoC FPGA architecture



Before - a USD 3000 development platform (hard to fit in carry-on luggage)

Now – Renode, a free and open source framework that’s in your PC (or server)


Get Renode™ for:

Debian Linux, (.deb) Download

Fedora Linux, (.rpm) Download

Arch Linux, (.pkg.tar.xz) Download

macOS Download

Windows Download

PFSoC support

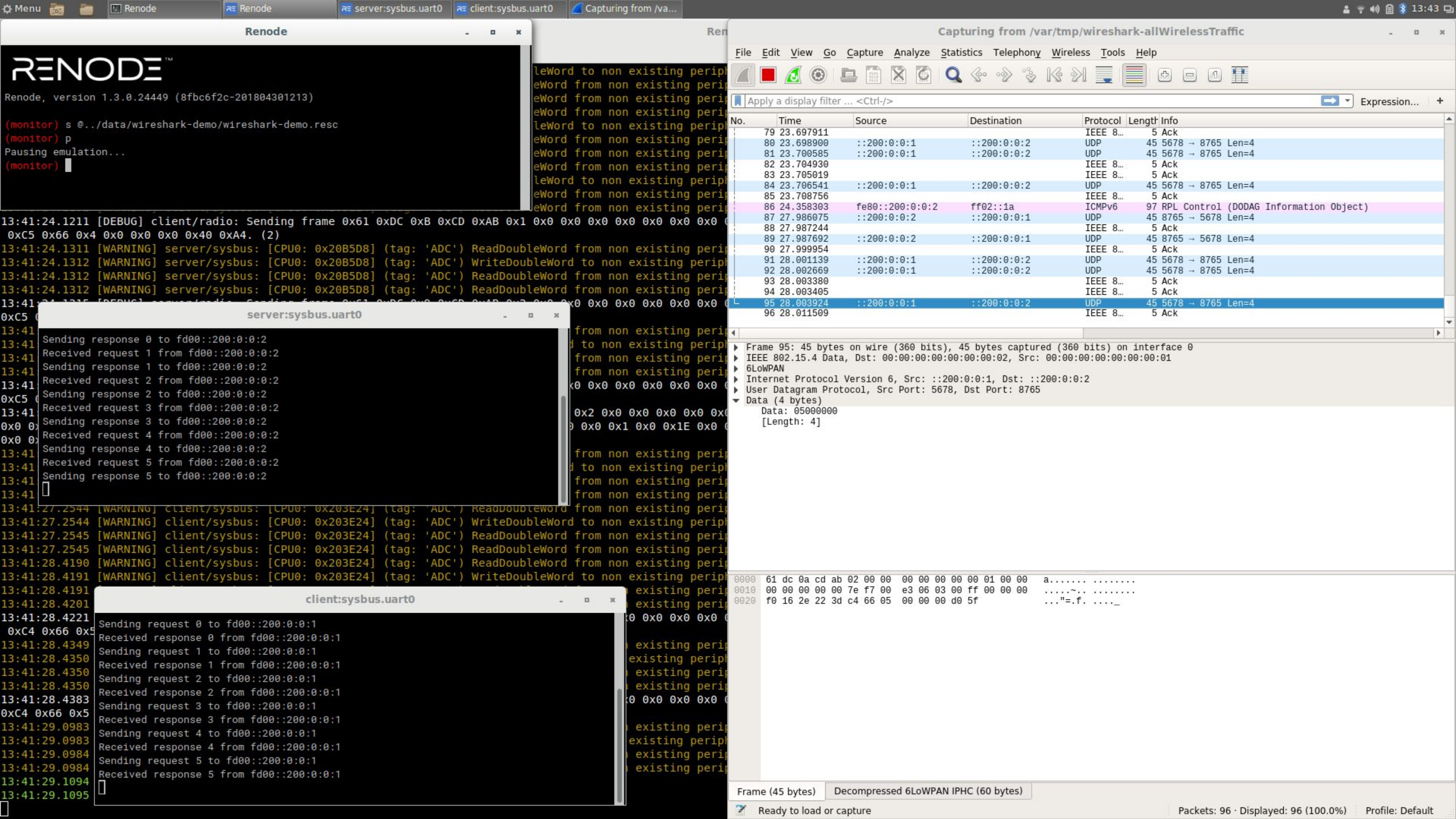
- Entire SoC complex
- include lots of I/O like USB, PCIe, CAN, I2C, SPI, GPIO, Eth...
- can model additional peripherals in the FPGA - easy to add new models as blocks
- integration with Verilator to actually co-simulate the IPs

PFSoC support highlights

- interfaces for multi-node connectivity
- also host-guest networking for Eth
- networking includes TSN/PTP
- analyze protocols, debug entire system at the same time

CAN

Ethernet



```
Renode, version 1.3.0.24449 (8fbc6f2c-201804301213)

(monitor) s @../data/wireshark-demo/wireshark-demo.resc
(monitor) p
Pausing emulation...
(monitor)
```

```
leWord to non existing periph
eWord from non existing perip
eWord from non existing perip
eWord from non existing perip
leWord to non existing periph
eWord from non existing perip
eWord from non existing perip
eWord from non existing perip
leWord to non existing periph
eWord from non existing perip
eWord from non existing perip
13:41:24.1211 [DEBUG] client/radio: Sending frame 0x61 0xDC 0xB 0xCD 0xAB 0x1 0x0 0x0 0x0 0x0 0x0 0x0
 0xC5 0x66 0x4 0x0 0x0 0x0 0x40 0xA4. (2)
13:41:24.1311 [WARNING] server/sysbus: [CPU0: 0x20B5D8] (tag: 'ADC') ReadDoubleWord from non existing perip
13:41:24.1312 [WARNING] server/sysbus: [CPU0: 0x20B5D8] (tag: 'ADC') WriteDoubleWord to non existing periph
13:41:24.1312 [WARNING] server/sysbus: [CPU0: 0x20B5D8] (tag: 'ADC') ReadDoubleWord from non existing perip
13:41:24.1312 [WARNING] server/sysbus: [CPU0: 0x20B5D8] (tag: 'ADC') ReadDoubleWord from non existing perip
13:41:27.2544 [WARNING] client/sysbus: [CPU0: 0x203E24] (tag: 'ADC') WriteDoubleWord to non existing periph
13:41:27.2545 [WARNING] client/sysbus: [CPU0: 0x203E24] (tag: 'ADC') ReadDoubleWord from non existing perip
13:41:27.2545 [WARNING] client/sysbus: [CPU0: 0x203E24] (tag: 'ADC') ReadDoubleWord from non existing perip
13:41:28.4190 [WARNING] client/sysbus: [CPU0: 0x203E24] (tag: 'ADC') ReadDoubleWord from non existing perip
13:41:28.4191 [WARNING] client/sysbus: [CPU0: 0x203E24] (tag: 'ADC') WriteDoubleWord to non existing periph
```

server:sysbus.uart0

```
Sending response 0 to fd00::200:0:0:2
Received request 1 from fd00::200:0:0:2
Sending response 1 to fd00::200:0:0:2
Received request 2 from fd00::200:0:0:2
Sending response 2 to fd00::200:0:0:2
Received request 3 from fd00::200:0:0:2
Sending response 3 to fd00::200:0:0:2
Received request 4 from fd00::200:0:0:2
Sending response 4 to fd00::200:0:0:2
Received request 5 from fd00::200:0:0:2
Sending response 5 to fd00::200:0:0:2
```

client:sysbus.uart0

```
Sending request 0 to fd00::200:0:0:1
Received response 0 from fd00::200:0:0:1
Sending request 1 to fd00::200:0:0:1
Received response 1 from fd00::200:0:0:1
Sending request 2 to fd00::200:0:0:1
Received response 2 from fd00::200:0:0:1
Sending request 3 to fd00::200:0:0:1
Received response 3 from fd00::200:0:0:1
Sending request 4 to fd00::200:0:0:1
Received response 4 from fd00::200:0:0:1
Sending request 5 to fd00::200:0:0:1
Received response 5 from fd00::200:0:0:1
```

Capturing from /var/tmp/wireshark-allWirelessTraffic

No.	Time	Source	Destination	Protocol	Length	Info
79	23.697911			IEEE 8…	5	Ack
80	23.698900	::200:0:0:1	::200:0:0:2	UDP	45	5678 → 8765 Len=4
81	23.700585	::200:0:0:1	::200:0:0:2	UDP	45	5678 → 8765 Len=4
82	23.704930			IEEE 8…	5	Ack
83	23.705019			IEEE 8…	5	Ack
84	23.706541	::200:0:0:1	::200:0:0:2	UDP	45	5678 → 8765 Len=4
85	23.708756			IEEE 8…	5	Ack
86	24.358303	fe80::200:0:0:2	ff02::1a	ICMPv6	97	RPL Control (DODAG Information Object)
87	27.986075	::200:0:0:2	::200:0:0:1	UDP	45	8765 → 5678 Len=4
88	27.987244			IEEE 8…	5	Ack
89	27.987692	::200:0:0:2	::200:0:0:1	UDP	45	8765 → 5678 Len=4
90	27.999954			IEEE 8…	5	Ack
91	28.001139	::200:0:0:1	::200:0:0:2	UDP	45	5678 → 8765 Len=4
92	28.002669	::200:0:0:1	::200:0:0:2	UDP	45	5678 → 8765 Len=4
93	28.003380			IEEE 8…	5	Ack
94	28.003405			IEEE 8…	5	Ack
95	28.003924	::200:0:0:1	::200:0:0:2	UDP	45	5678 → 8765 Len=4
96	28.011509			IEEE 8…	5	Ack

```
▸ Frame 95: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface 0
▸ IEEE 802.15.4 Data, Dst: 00:00:00:00:00:00:00:02, Src: 00:00:00:00:00:00:00:01
▸ 6LoWPAN
▸ Internet Protocol Version 6, Src: ::200:0:0:1, Dst: ::200:0:0:2
▸ User Datagram Protocol, Src Port: 5678, Dst Port: 8765
▾ Data (4 bytes)
    Data: 05000000
    [Length: 4]
```

```
0000  61 dc 0a cd ab 02 00 00  00 00 00 00 00 01 00 00   a....... ........
0010  00 00 00 00 00 7e f7 00  e3 06 03 00 ff 00 00 00   .....~.. ........
0020  f0 16 2e 22 3d c4 66 05  00 00 00 d0 5f            ..."=.f. ...._
```

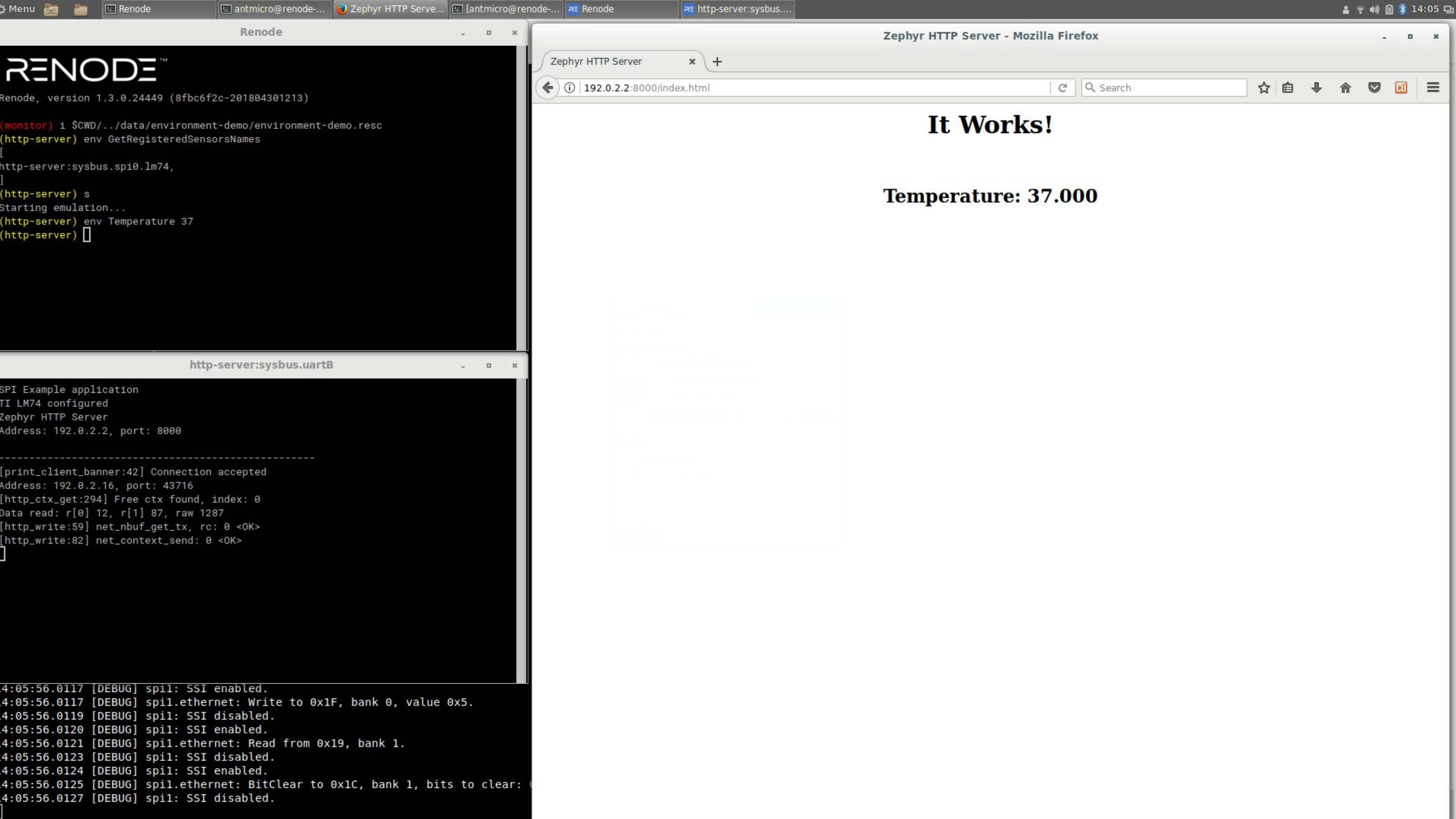
Frame (45 bytes)
Decompressed 6LoWPAN IPHC (60 bytes)
Ready to load or capture
Packets: 96 · Displayed: 96 (100.0%)
Profile: Default

PFSoC support highlights

- interfaces for connecting e.g. sensors and actuators
- quite useful for building real boards that interact with the external world

I2C

SPI



```
Renode, version 1.3.0.24449 (8fbc6f2c-201804301213)

(monitor) i $CWD/../data/environment-demo/environment-demo.resc
(http-server) env GetRegisteredSensorsNames
[
http-server:sysbus.spi0.lm74,
]
(http-server) s
Starting emulation...
(http-server) env Temperature 37
(http-server)
```

```
SPI Example application
TI LM74 configured
Zephyr HTTP Server
Address: 192.0.2.2, port: 8000

--------------------------------------------------
[print_client_banner:42] Connection accepted
Address: 192.0.2.16, port: 43716
[http_ctx_get:294] Free ctx found, index: 0
Data read: r[0] 12, r[1] 87, raw 1287
[http_write:59] net_nbuf_get_tx, rc: 0 <OK>
[http_write:82] net_context_send: 0 <OK>
```

```
4:05:56.0117 [DEBUG] spi1: SSI enabled.
4:05:56.0117 [DEBUG] spi1.ethernet: Write to 0x1F, bank 0, value 0x5.
4:05:56.0119 [DEBUG] spi1: SSI disabled.
4:05:56.0120 [DEBUG] spi1: SSI enabled.
4:05:56.0121 [DEBUG] spi1.ethernet: Read from 0x19, bank 1.
4:05:56.0123 [DEBUG] spi1: SSI disabled.
4:05:56.0124 [DEBUG] spi1: SSI enabled.
4:05:56.0125 [DEBUG] spi1.ethernet: BitClear to 0x1C, bank 1, bits to clear:
4:05:56.0127 [DEBUG] spi1: SSI disabled.
```

It Works!

Temperature: 37.000

PFSoC support highlights

- interfaces for connecting with interesting external elements
- enable to really explore the flexibility of Renode

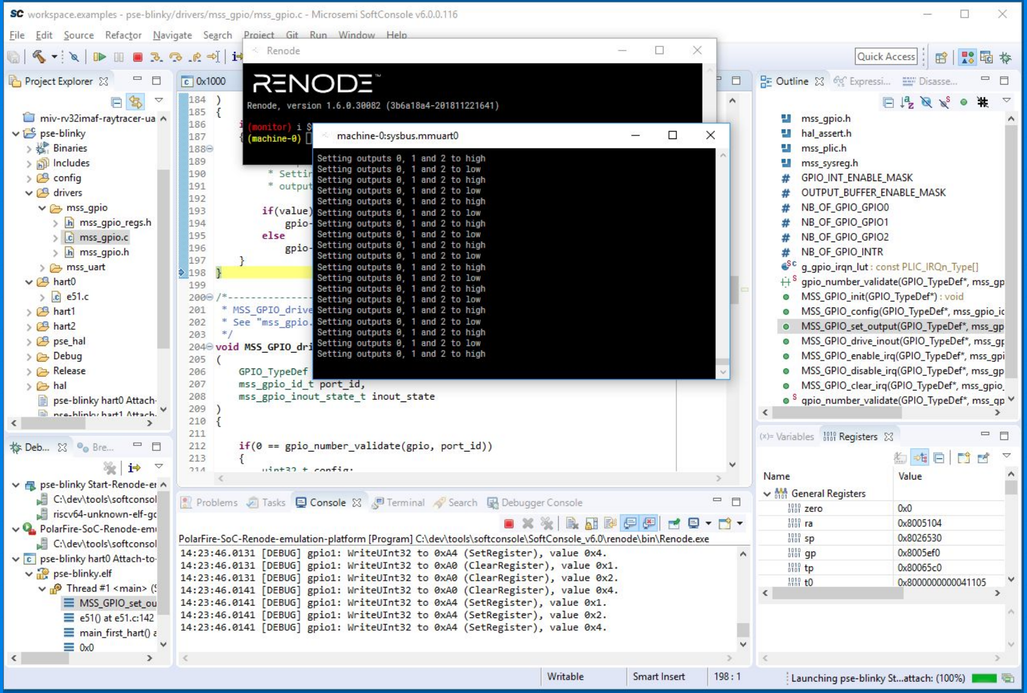
PCIe

USB

RENODE™

SoftConsole integration

- Standard IDE, comes bundled
- Linux and Windows
- examine the entire system as you're developing code
- new and exciting abilities


workspace.examples - pse-blinky/drivers/mss_gpio/mss_gpio.c - Microsemi SoftConsole v6.0.0.116

```
Renode, version 1.6.0.30082 (3b6a18a4-201811221641)

(monitor) i $
(machine-0)
```

machine-0:sysbus.mmuart0

```
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
```

```
    GPIO_TypeDef
    mss_gpio_id_t port_id,
    mss_gpio_inout_state_t inout_state
)
{

    if(0 == gpio_number_validate(gpio, port_id))
    {
```

PolarFire-SoC-Renode-emulation-platform [Program] C:\dev\tools\softconsole\SoftConsole_v6.0\renode\bin\Renode.exe

```
14:23:46.0131 [DEBUG] gpio1: WriteUInt32 to 0xA4 (SetRegister), value 0x4.
14:23:46.0131 [DEBUG] gpio1: WriteUInt32 to 0xA0 (ClearRegister), value 0x1.
14:23:46.0131 [DEBUG] gpio1: WriteUInt32 to 0xA0 (ClearRegister), value 0x2.
14:23:46.0141 [DEBUG] gpio1: WriteUInt32 to 0xA0 (ClearRegister), value 0x4.
14:23:46.0141 [DEBUG] gpio1: WriteUInt32 to 0xA4 (SetRegister), value 0x1.
14:23:46.0141 [DEBUG] gpio1: WriteUInt32 to 0xA4 (SetRegister), value 0x2.
14:23:46.0141 [DEBUG] gpio1: WriteUInt32 to 0xA4 (SetRegister), value 0x4.
```

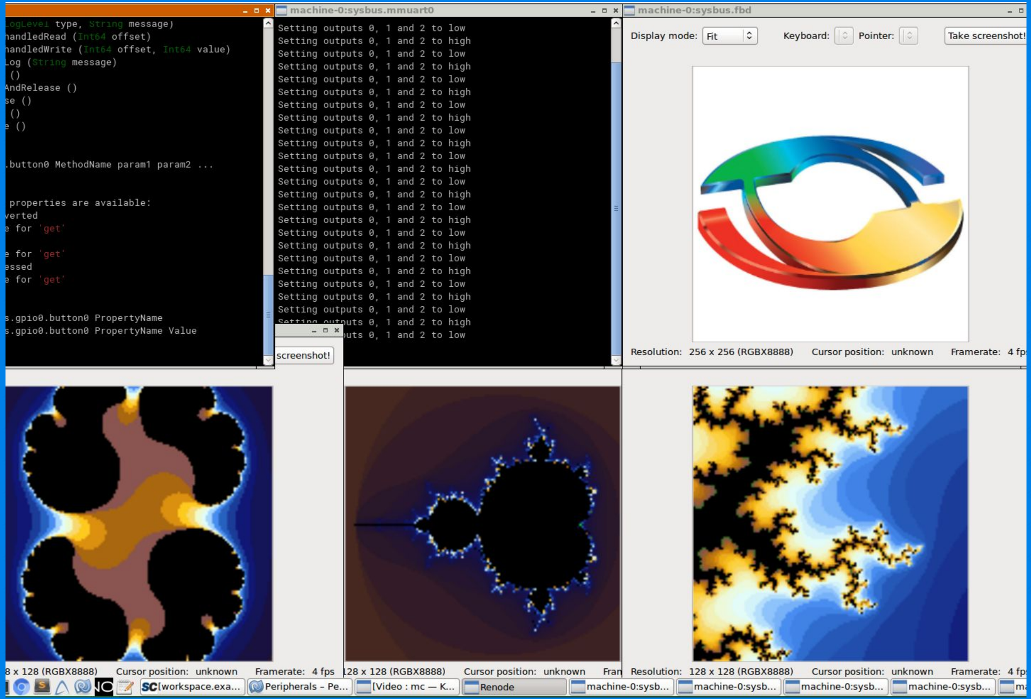
Name	Value
General Registers	
zero	0x0
ra	0x8005104
sp	0x8026530
gp	0x8005ef0
tp	0x80065c0
t0	0x8000000000041105

Launching pse-blinky St...attach: (100%)

RENODE™

SoftConsole integration

- Renode is extremely extendible
- Debug, tracing, visualisation - we have all the data



```
LogLevel type, String message)
handledRead (Int64 offset)
handledWrite (Int64 offset, Int64 value)
Log (String message)
 ()
AndRelease ()
se ()
 ()
e ()


.button0 MethodName param1 param2 ...


 properties are available:
verted
e for 'get'

e for 'get'
essed
e for 'get'


s.gpio0.button0 PropertyName
s.gpio0.button0 PropertyName Value
```

machine-0:sysbus.mmuart0

```
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
        puts 0, 1 and 2 to low
```

machine-0:sysbus.fbd

Display mode: Fit   Keyboard: Pointer: Take screenshot!

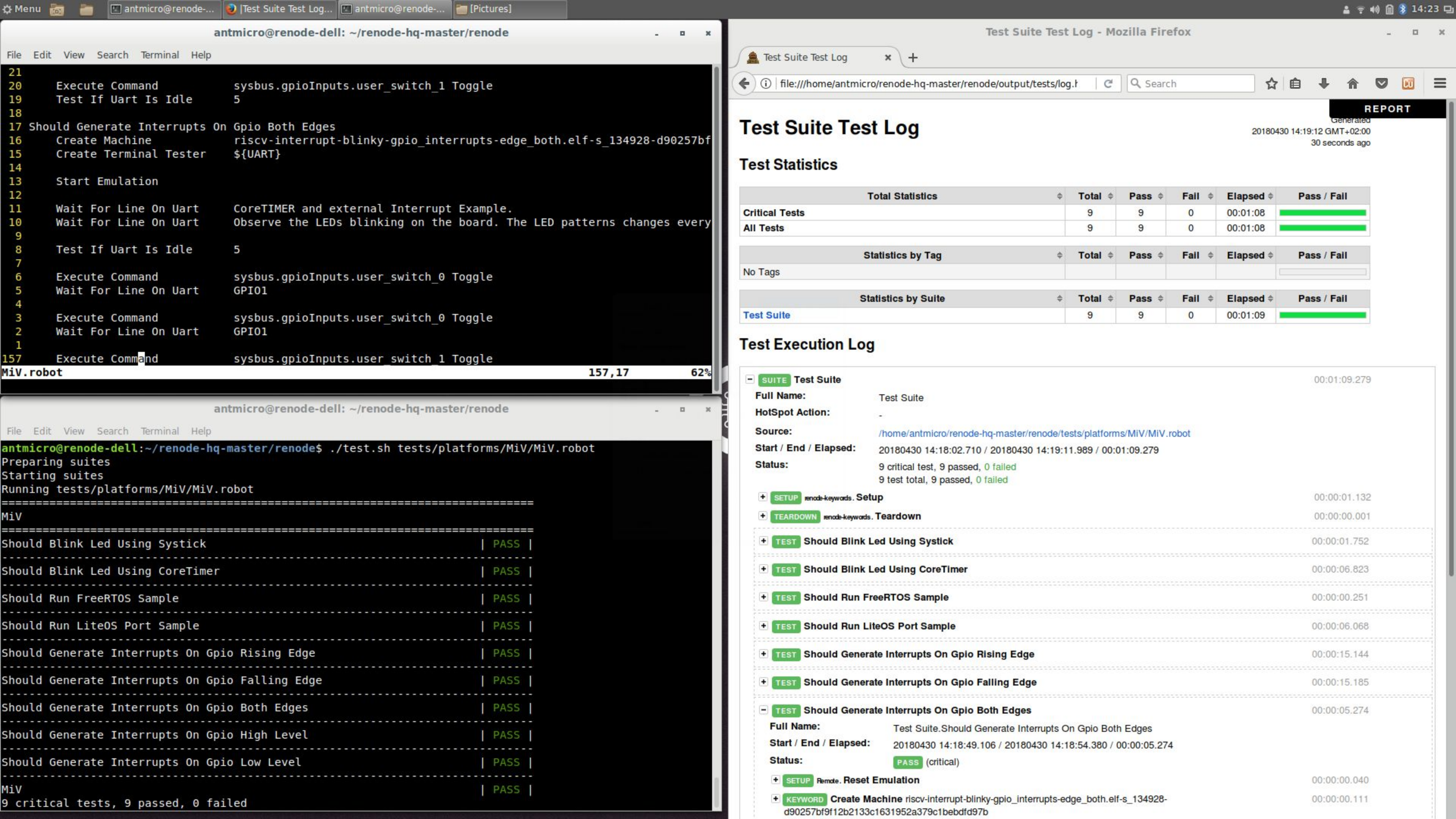
Resolution: 256 x 256 (RGBX8888)   Cursor position: unknown   Framerate: 4 fps

8 x 128 (RGBX8888)   Cursor position: unknown   Framerate: 4 fps

28 x 128 (RGBX8888)   Cursor position: unknown   Fran

Resolution: 128 x 128 (RGBX8888)   Cursor position: unknown   Framerate: 4 fp

By: antmicro EMBEDDED SYSTEMS



Menu
antmicro@renode-...
Test Suite Test Log...
antmicro@renode-...
[Pictures]
14:23
antmicro@renode-dell: ~/renode-hq-master/renode
File Edit View Search Terminal Help

```

    Execute Command          sysbus.gpioInputs.user_switch_1 Toggle
    Test If Uart Is Idle     5

Should Generate Interrupts On Gpio Both Edges
    Create Machine           riscv-interrupt-blinky-gpio_interrupts-edge_both.elf-s_134928-d90257bf
    Create Terminal Tester   ${UART}

    Start Emulation

    Wait For Line On Uart    CoreTIMER and external Interrupt Example.
    Wait For Line On Uart    Observe the LEDs blinking on the board. The LED patterns changes every

    Test If Uart Is Idle     5

    Execute Command          sysbus.gpioInputs.user_switch_0 Toggle
    Wait For Line On Uart    GPIO1

    Execute Command          sysbus.gpioInputs.user_switch_0 Toggle
    Wait For Line On Uart    GPIO1

    Execute Command          sysbus.gpioInputs.user_switch_1 Toggle
MiV.robot                                                        157,17          62%
```

antmicro@renode-dell: ~/renode-hq-master/renode
File Edit View Search Terminal Help

```
antmicro@renode-dell:~/renode-hq-master/renode$ ./test.sh tests/platforms/MiV/MiV.robot
Preparing suites
Starting suites
Running tests/platforms/MiV/MiV.robot
==============================================================================
MiV
==============================================================================
Should Blink Led Using Systick                                        | PASS |
------------------------------------------------------------------------------
Should Blink Led Using CoreTimer                                      | PASS |
------------------------------------------------------------------------------
Should Run FreeRTOS Sample                                            | PASS |
------------------------------------------------------------------------------
Should Run LiteOS Port Sample                                         | PASS |
------------------------------------------------------------------------------
Should Generate Interrupts On Gpio Rising Edge                        | PASS |
------------------------------------------------------------------------------
Should Generate Interrupts On Gpio Falling Edge                       | PASS |
------------------------------------------------------------------------------
Should Generate Interrupts On Gpio Both Edges                         | PASS |
------------------------------------------------------------------------------
Should Generate Interrupts On Gpio High Level                         | PASS |
------------------------------------------------------------------------------
Should Generate Interrupts On Gpio Low Level                          | PASS |
------------------------------------------------------------------------------
MiV                                                                   | PASS |
9 critical tests, 9 passed, 0 failed
```

Test Suite Test Log - Mozilla Firefox
Test Suite Test Log
file:///home/antmicro/renode-hq-master/renode/output/tests/log.h
Search
REPORT

Test Suite Test Log

Generated
20180430 14:19:12 GMT+02:00
30 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	9	9	0	00:01:08	🟩
All Tests	9	9	0	00:01:08	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Test Suite	9	9	0	00:01:09	🟩

Test Execution Log

```
− SUITE Test Suite                                                    00:01:09.279
  Full Name:            Test Suite
  HotSpot Action:       -
  Source:               /home/antmicro/renode-hq-master/renode/tests/platforms/MiV/MiV.robot
  Start / End / Elapsed: 20180430 14:18:02.710 / 20180430 14:19:11.989 / 00:01:09.279
  Status:               9 critical test, 9 passed, 0 failed
                        9 test total, 9 passed, 0 failed
  + SETUP renode-keywords. Setup                                      00:00:01.132
  + TEARDOWN renode-keywords. Teardown                                00:00:00.001
  + TEST Should Blink Led Using Systick                               00:00:01.752
  + TEST Should Blink Led Using CoreTimer                             00:00:06.823
  + TEST Should Run FreeRTOS Sample                                   00:00:00.251
  + TEST Should Run LiteOS Port Sample                                00:00:06.068
  + TEST Should Generate Interrupts On Gpio Rising Edge               00:00:15.144
  + TEST Should Generate Interrupts On Gpio Falling Edge              00:00:15.185
  − TEST Should Generate Interrupts On Gpio Both Edges                00:00:05.274
    Full Name:            Test Suite.Should Generate Interrupts On Gpio Both Edges
    Start / End / Elapsed: 20180430 14:18:49.106 / 20180430 14:18:54.380 / 00:00:05.274
    Status:               PASS (critical)
    + SETUP Remote. Reset Emulation                                   00:00:00.040
    + KEYWORD Create Machine riscv-interrupt-blinky-gpio_interrupts-edge_both.elf-s_134928-
      d90257bf9f12b2133c1631952a379c1bebdfd97b                        00:00:00.111
```

Significance of PolarFire SoC platform

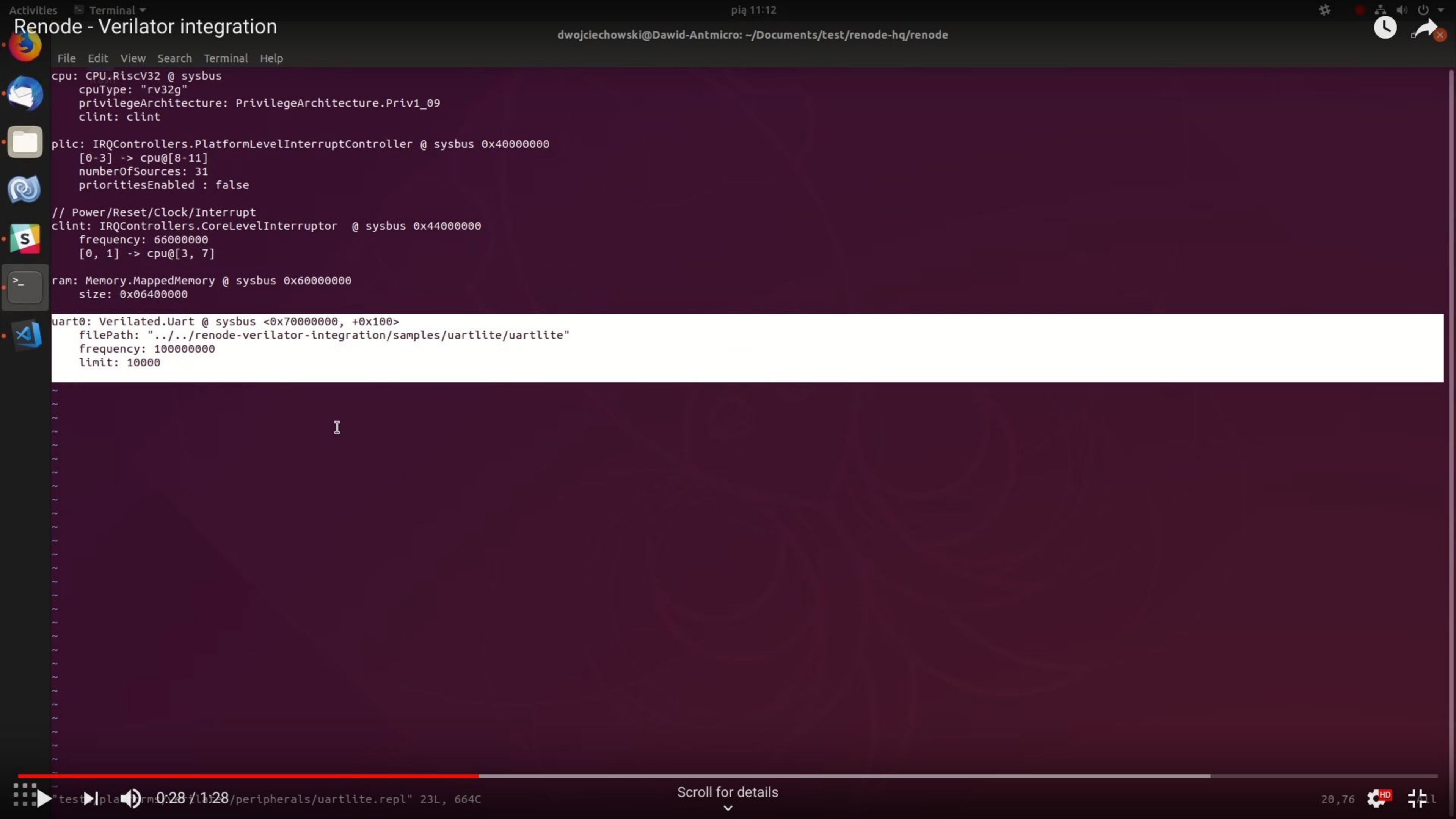
- as an FPGA SoC, ideal for developing and prototyping new RISC-V hardware solutions
- custom accelerators in FPGA fabric
- Linux + real-time systems
- showing the way with an ultra-flexible **pre-silicon** development platform for all developers - Renode

Microsemi
a Microchip company

antmicro

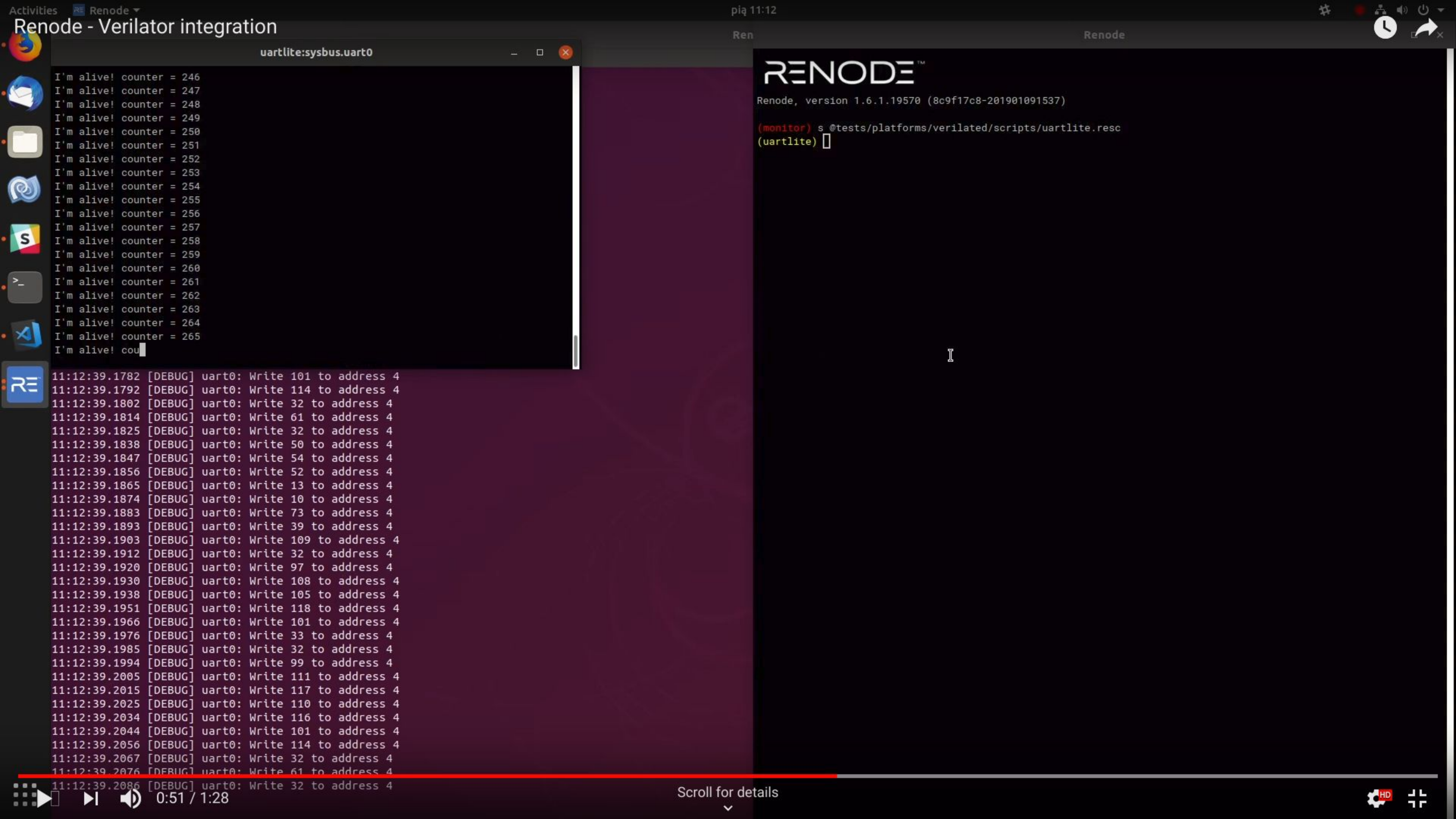
Co-simulation with Verilator

- For IP you care about, you can simulate the real RTL with Verilator
- The co-simulated block works as a regular Renode block (of course much slower), driven from Renode with all APIs, only a small shim needed
- Divide and conquer your problem - simulate most of the system fast and limit HDL simulation to minimum
- Developing and extending this feature with multiple partners


Renode - Verilator integration

dwojciechowski@Dawid-Antmicro: ~/Documents/test/renode-hq/renode

```
cpu: CPU.RiscV32 @ sysbus
    cpuType: "rv32g"
    privilegeArchitecture: PrivilegeArchitecture.Priv1_09
    clint: clint

plic: IRQControllers.PlatformLevelInterruptController @ sysbus 0x40000000
    [0-3] -> cpu@[8-11]
    numberOfSources: 31
    prioritiesEnabled : false

// Power/Reset/Clock/Interrupt
clint: IRQControllers.CoreLevelInterruptor  @ sysbus 0x44000000
    frequency: 66000000
    [0, 1] -> cpu@[3, 7]

ram: Memory.MappedMemory @ sysbus 0x60000000
    size: 0x06400000

uart0: Verilated.Uart @ sysbus <0x70000000, +0x100>
    filePath: "../../renode-verilator-integration/samples/uartlite/uartlite"
    frequency: 100000000
    limit: 10000
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"test/pla  rms/  /peripherals/uartlite.repl" 23L, 664C                 20,76
```



Renode - Verilator integration

uartlite:sysbus.uart0

```
I'm alive! counter = 246
I'm alive! counter = 247
I'm alive! counter = 248
I'm alive! counter = 249
I'm alive! counter = 250
I'm alive! counter = 251
I'm alive! counter = 252
I'm alive! counter = 253
I'm alive! counter = 254
I'm alive! counter = 255
I'm alive! counter = 256
I'm alive! counter = 257
I'm alive! counter = 258
I'm alive! counter = 259
I'm alive! counter = 260
I'm alive! counter = 261
I'm alive! counter = 262
I'm alive! counter = 263
I'm alive! counter = 264
I'm alive! counter = 265
I'm alive! cou
```

```
11:12:39.1782 [DEBUG] uart0: Write 101 to address 4
11:12:39.1792 [DEBUG] uart0: Write 114 to address 4
11:12:39.1802 [DEBUG] uart0: Write 32 to address 4
11:12:39.1814 [DEBUG] uart0: Write 61 to address 4
11:12:39.1825 [DEBUG] uart0: Write 32 to address 4
11:12:39.1838 [DEBUG] uart0: Write 50 to address 4
11:12:39.1847 [DEBUG] uart0: Write 54 to address 4
11:12:39.1856 [DEBUG] uart0: Write 52 to address 4
11:12:39.1865 [DEBUG] uart0: Write 13 to address 4
11:12:39.1874 [DEBUG] uart0: Write 10 to address 4
11:12:39.1883 [DEBUG] uart0: Write 73 to address 4
11:12:39.1893 [DEBUG] uart0: Write 39 to address 4
11:12:39.1903 [DEBUG] uart0: Write 109 to address 4
11:12:39.1912 [DEBUG] uart0: Write 32 to address 4
11:12:39.1920 [DEBUG] uart0: Write 97 to address 4
11:12:39.1930 [DEBUG] uart0: Write 108 to address 4
11:12:39.1938 [DEBUG] uart0: Write 105 to address 4
11:12:39.1951 [DEBUG] uart0: Write 118 to address 4
11:12:39.1966 [DEBUG] uart0: Write 101 to address 4
11:12:39.1976 [DEBUG] uart0: Write 33 to address 4
11:12:39.1985 [DEBUG] uart0: Write 32 to address 4
11:12:39.1994 [DEBUG] uart0: Write 99 to address 4
11:12:39.2005 [DEBUG] uart0: Write 111 to address 4
11:12:39.2015 [DEBUG] uart0: Write 117 to address 4
11:12:39.2025 [DEBUG] uart0: Write 110 to address 4
11:12:39.2034 [DEBUG] uart0: Write 116 to address 4
11:12:39.2044 [DEBUG] uart0: Write 101 to address 4
11:12:39.2056 [DEBUG] uart0: Write 114 to address 4
11:12:39.2067 [DEBUG] uart0: Write 32 to address 4
11:12:39.2076 [DEBUG] uart0: Write 61 to address 4
11:12:39.2086 [DEBUG] uart0: Write 32 to address 4
```

Renode

```
Renode, version 1.6.1.19570 (8c9f17c8-201901091537)

(monitor) s @tests/platforms/verilated/scripts/uartlite.resc
(uartlite)
```

Example: LiteX soft SoC

- open source configurable SoC, RISC-V option
- runs 32-bit Linux now!
- Renode model of core and peripherals e.g. Ethernet, UART etc.
- can build, for example, a simulated setup with multiple Ethernets
- easily add extra HDL peripherals through Verilator integration

HW/SW co-development: Dover Microsystems

DOVER MICROSYSTEMS

- Dover is developing CoreGuard™ cybersecurity silicon IP, used by customers such as NXP
- Renode was extended by Antmicro for Dover with fine-grained control of execution and debugging
- Dover uses Renode both in internal development as well as externally

HW/SW co-development: Dover Microsystems - uses

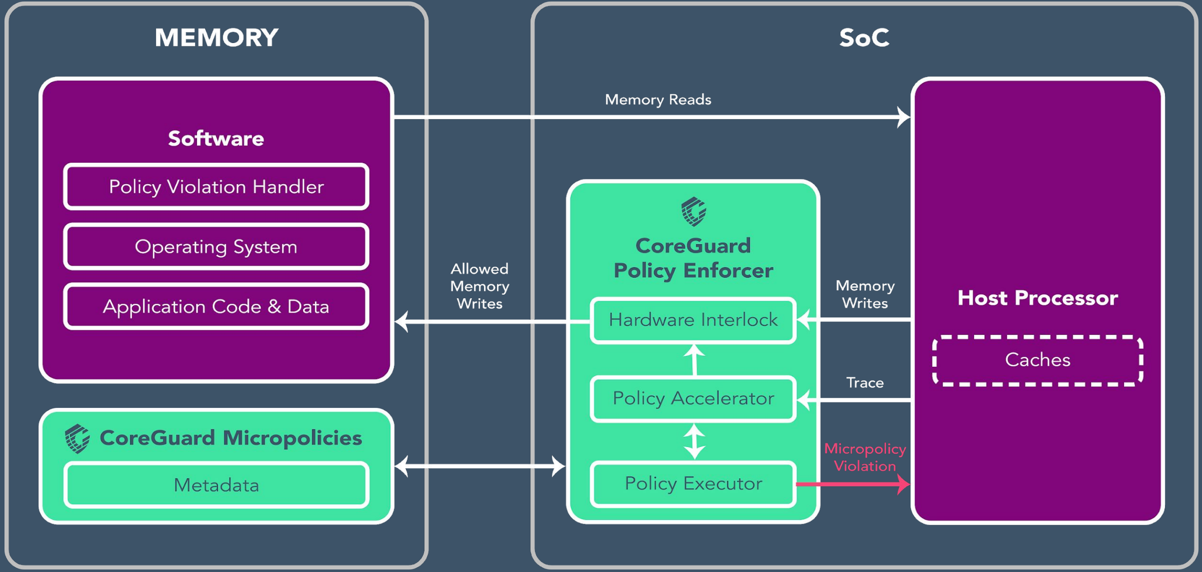
DOVER MICROSYSTEMS

- Design space exploration
- Implementation
- Testing
- Demonstration and evaluation

COREGUARD - SECURITY ENFORCEMENT IN HW

Architecture agnostic IP licensed and delivered as **hardware design files**

- Integrates with RISC processors to provide separate, sentry logic.
- Extracts a set of trace signals from host processor
- Monitors and evaluates every instruction in the host processor in real time
- Provide mechanism for CoreGuard to affect a stall on the host when needed to evaluate policies
- Exceptions thrown from CoreGuard when policy violated



MICROPOLICIES & METADATA

A set of rules that express the allowed combinations of metadata for each possible CPU operation.

FOCUS	MICROPOLICY EXAMPLES		
SECURITY	**Heap Protection**	**Stack Protection**	Globals Protection
	RWX (Read, Write, Execute)	Data Type Enforcement	Procedure Enforcement
	Control Flow Integrity	Fine-Grained Access Control	Sandbox
	Compartmentalization	Code Protect	Resource Management
PRIVACY	Information Flow	Multi-Level Classification	Data Privacy
SAFETY	Medical	Automotive	FSA Enforcement

METADATA

CoreGuard uses the metadata information today's processors have been throwing away.

Metadata defines …

if the value is **private**

if the value is **a pointer**

what a pointer has **access** to

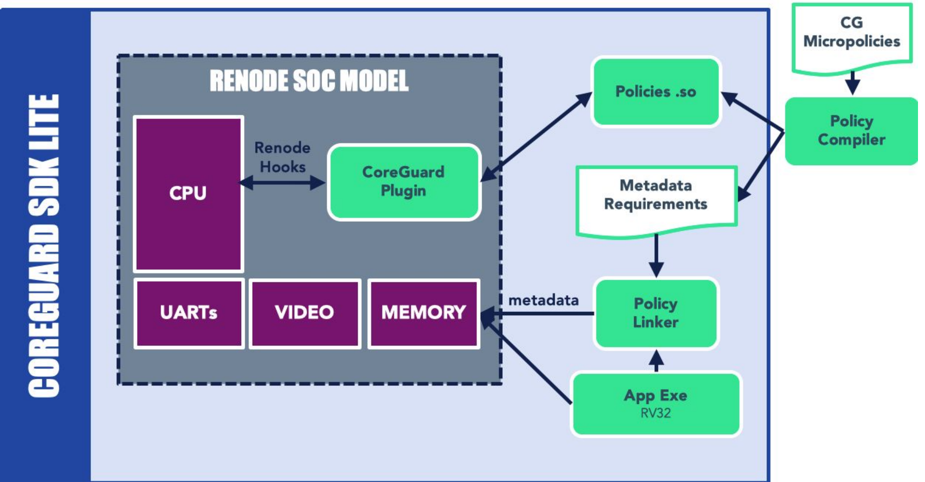
if the data is **executable**

if the reference word in memory is a **return address**

TWO CONTEXTS OF DOVER USING RENODE

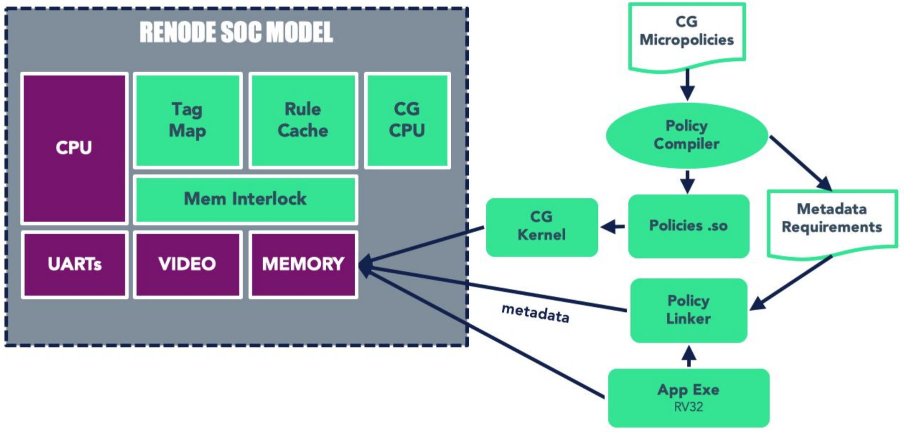
CUSTOMER DELIVERABLE

- SoC has AP CPU + peripherals
- CoreGuard is entirely simulated in C# and C++
  - Policy code runs on host (e.g. X86)
- Via a Renode Plugin
  - Registers BlockBeginHook and BlockEndHook Renode hooks
  - Hooks call generated C++ code



INTERNAL DEVELOPMENT

- SoC has AP CPU + peripherals + CG PEX (RISC-V CPU), CG "accelerator"
- Policy code runs on simulated RISC-V
- Used to debug boot and runtime CG API



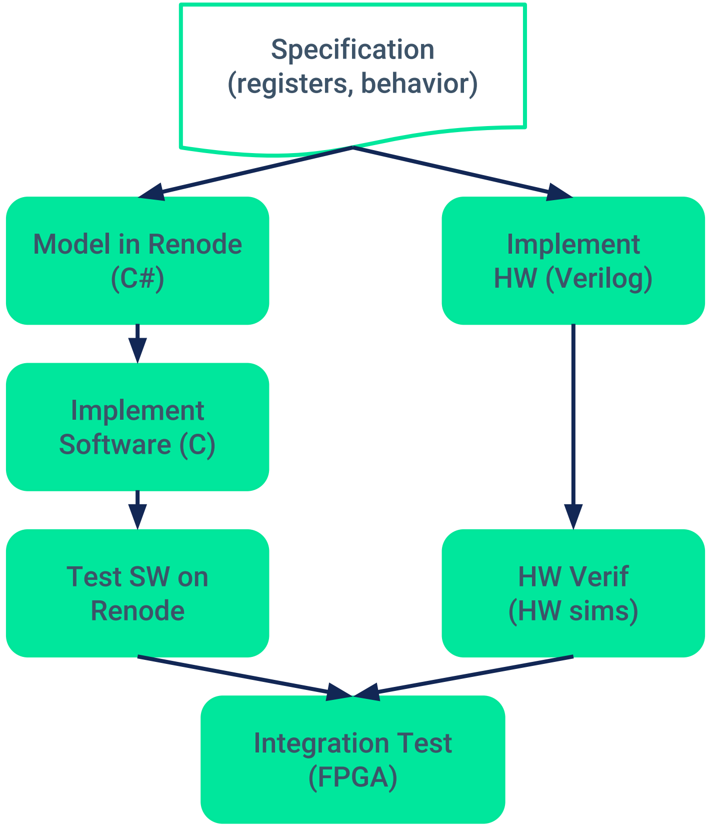
PARALLEL HW/SW DEVELOPMENT

Use Case

- HW spec developed between HW and SW teams
- SW team implements spec in Renode and writes firmware against spec, testing on Renode
- In parallel, HW is implementing and testing HW design
- Integration test via FPGA and/or HW simulator

EXAMPLE

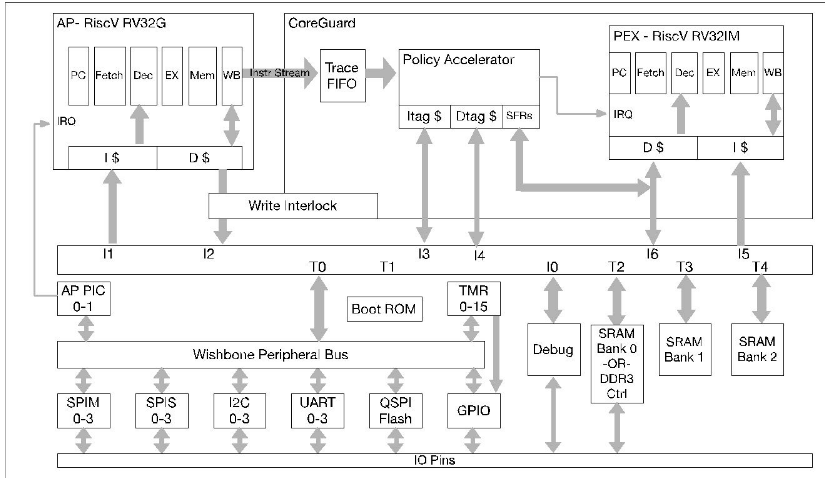
HDMI device. We had SW working against spec, under Renode, in advance of HW.



TESTING BOOT CODE

Use Case

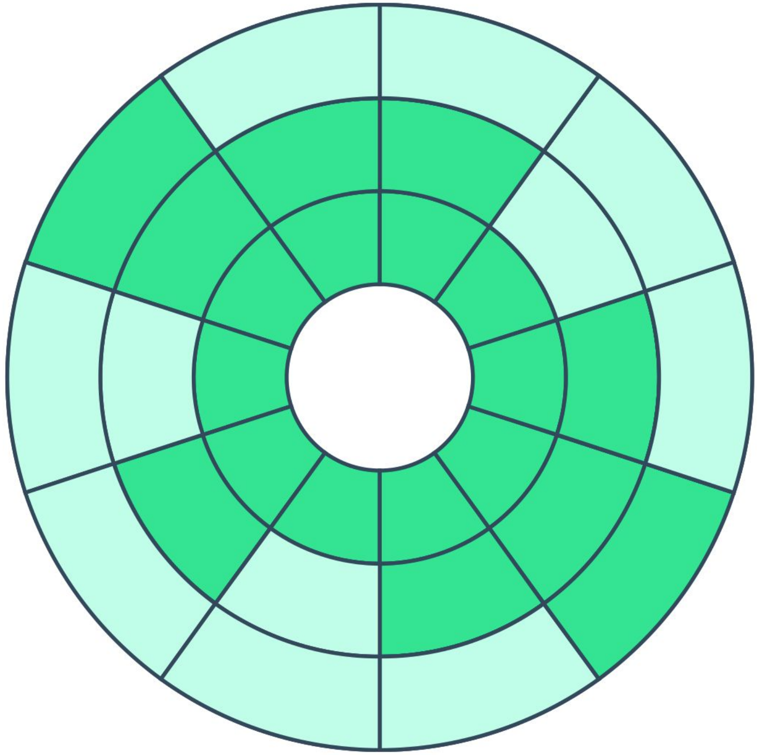
- Non ELF formats (scatter-load) used by HW boot ROMs
  - Easy to test in Renode
- Need full SoC simulated; copying from flash to RAM, initializing devices
  - Can instrument Renode to flag erroneous bus traffic
  - Full debugging under Renode (rarely the case at boot time on HW)
- Peripherals (UARTs, PICs, our CoreGuard interface) can be instrumented to check for correct initialization and use
- Playing with memory map



PROFILING

Use Case

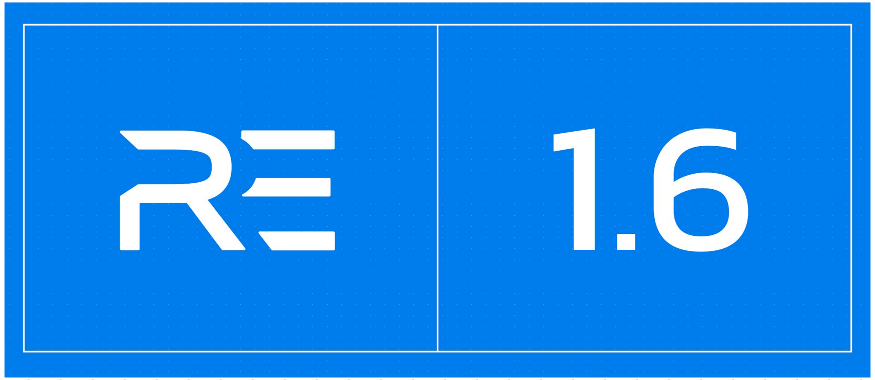
- *Before* hardware design, can profile different cache hierarchies, data representations, etc.
- Get "macro" numbers – cache hits vs. misses, number of indirections (memory pressure), etc.



Dover – effects

- Renode allows Dover to run thousands of tests daily, preventing regressions (particularly bad for security IP)
- Decreased turnaround for new feature prototyping from days/weeks to hours
- Enabled customers to easily test-drive the technology

Interested? Just get it from GitHub!



... and be sure to see our demo + talk to us

RENODE™

Renode development services

Embedded systems development services powered by the RENODE methodology

Professional support, implementing new platforms

Building customized tools, user interfaces and integrations

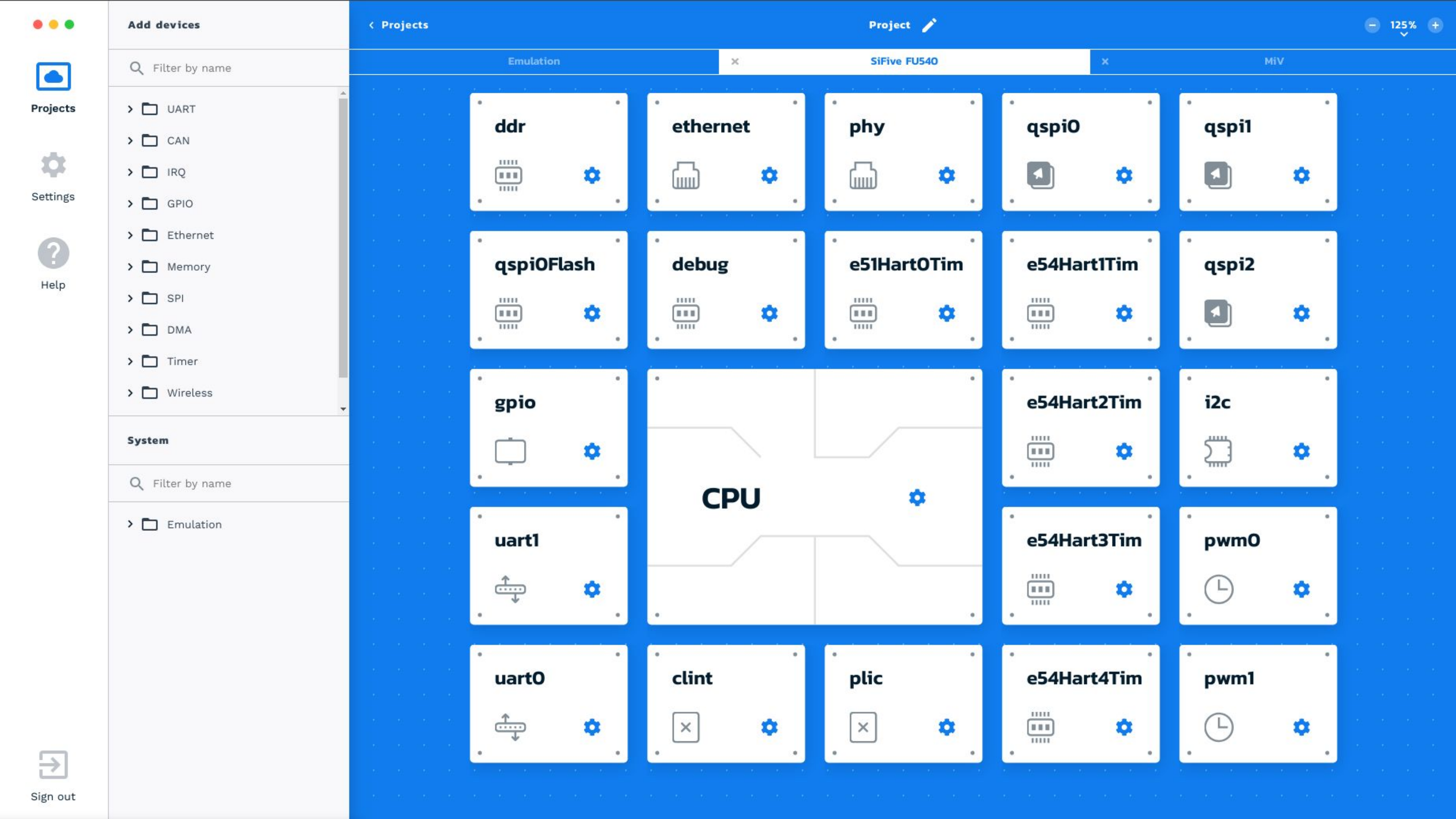
Setting up CI and improving development workflows in your organization

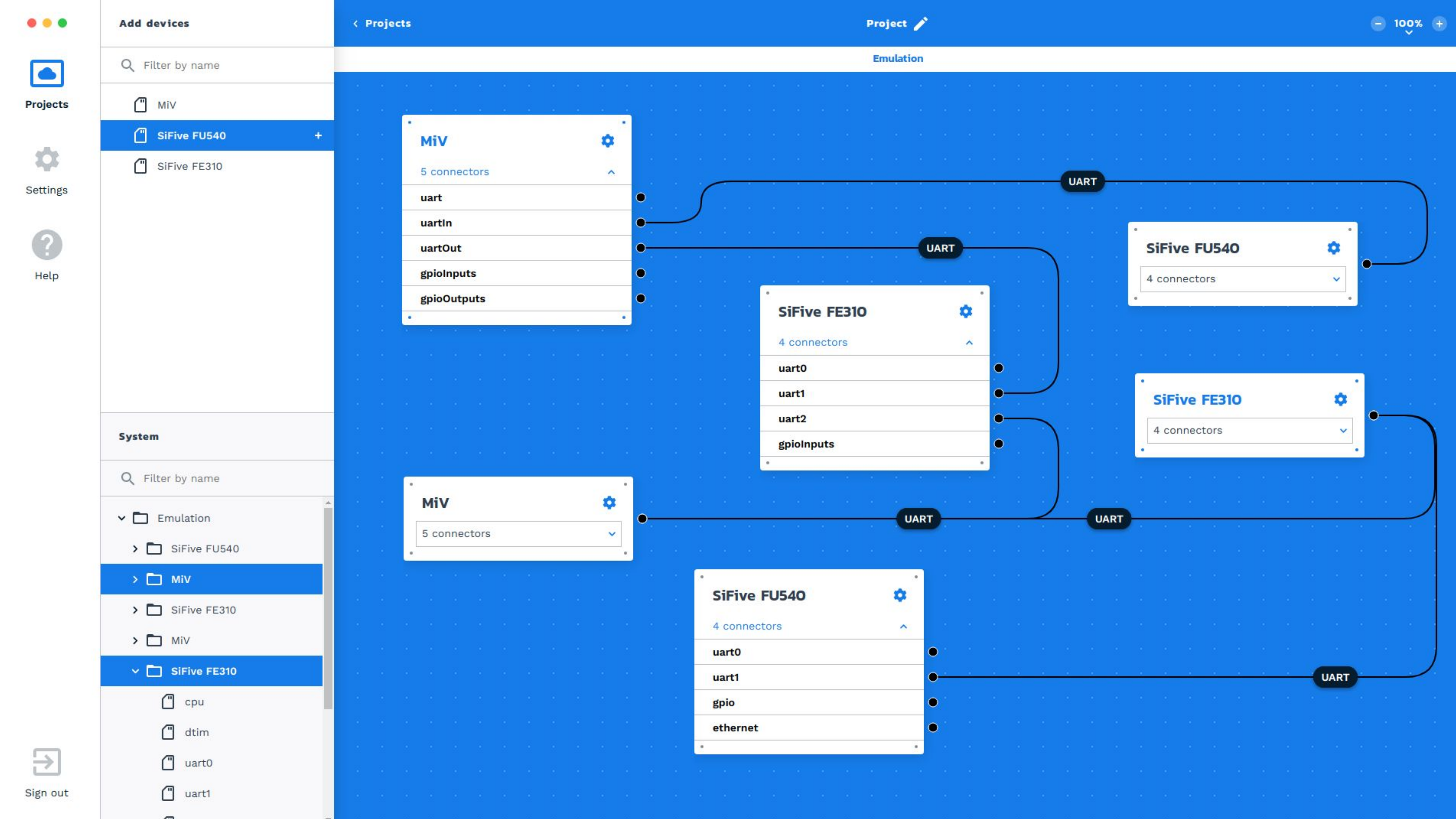
By: antmicro

Summary

- Renode allows practical HW/SW co-design on many levels
- Its flexibility enables easy prototyping and practical adoption for RISC-V & mixed systems
- Microsemi (Microchip) has shown the way for enabling users with pre-silicon development
- Features like Verilator integration are designed to further strengthen the HW/SW co-design use-case
- Dover case shows how you can use Renode throughout the entire development cycle

A sneak peek into the future


Projects
Settings
Help
Sign out
Add devices
Filter by name
UART
CAN
IRQ
GPIO
Ethernet
Memory
SPI
DMA
Timer
Wireless
System
Filter by name
Emulation
Projects
Project
125%
Emulation
SiFive FU540
MiV
ddr
ethernet
phy
qspi0
qspi1
qspi0Flash
debug
e51Hart0Tim
e54Hart1Tim
qspi2
gpio
CPU
e54Hart2Tim
i2c
uart1
e54Hart3Tim
pwm0
uart0
clint
plic
e54Hart4Tim
pwm1


Projects
Settings
Help
Sign out
Add devices
Filter by name
MiV
SiFive FU540
SiFive FE310
System
Filter by name
Emulation
SiFive FU540
MiV
SiFive FE310
MiV
SiFive FE310
cpu
dtim
uart0
uart1
Projects
Project
100%
Emulation
MiV
5 connectors
uart
uartIn
uartOut
gpioInputs
gpioOutputs
UART
UART
SiFive FU540
4 connectors
SiFive FE310
4 connectors
uart0
uart1
uart2
gpioInputs
SiFive FE310
4 connectors
MiV
5 connectors
UART
UART
SiFive FU540
4 connectors
uart0
uart1
gpio
ethernet
UART