Styx Emulator

Public Release Presentation and Roadmap

Jordan Moore (lockbox)

RSTCON
Savannah, GA 2025

The Vision

- A uniform way to programmatically emulate and model systems
- A single emulation and modeling API to build with
- A unification of tools that already exist so you don't lose the work already invested
- A flexible emulator framework built to be tailored by users, not just developers

The Vision (cont.)

- Connect a … to an emulator
  - Physics Simulator / MATLAB
  - Fuzzer / GNURadio
  - Physical Hardware / Sensor
  - $custom_thing
  - Another emulator

The Vision (cont.)

- Connect a … to an emulator
  - Physics Simulator / MATLAB
  - Fuzzer / GNURadio
  - Physical Hardware / Sensor
  - $custom_thing
  - Another emulator
- Using the same tool

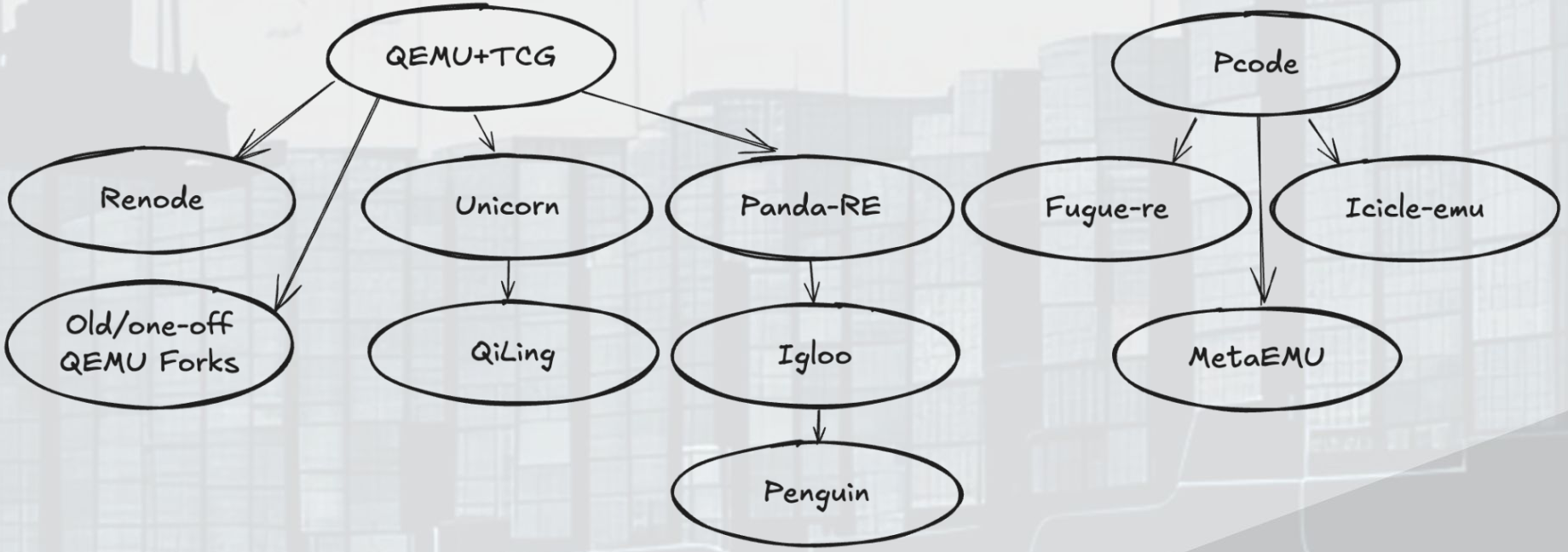
The Vision (cont.)

- Connect a … to an emulator
  - Physics Simulator / MATLAB
  - Fuzzer / GNURadio
  - Physical Hardware / Sensor
  - $custom_thing
  - Another emulator
- Using the same tool
- Be able to add custom support
  - New Architecture
  - New Chip
  - New Peripheral
  - $custom_thing

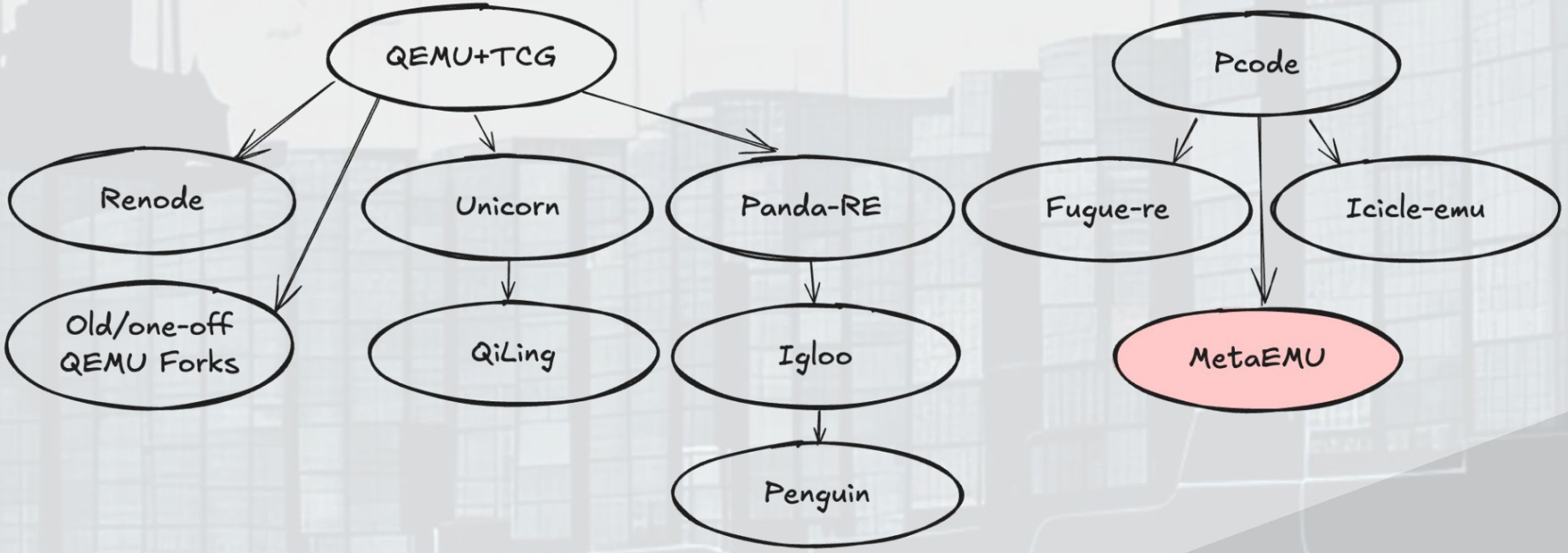
The Vision (cont.)

- Connect a … to an emulator
  - Physics Simulator / MATLAB
  - Fuzzer / GNURadio
  - Physical Hardware / Sensor
  - $custom_thing
  - Another emulator
- Using the same tool
- Be able to add custom support
  - New Architecture
  - New Chip
  - New Peripheral
  - $custom_thing
- Quickly, every time

Open Source Emulator Family Tree

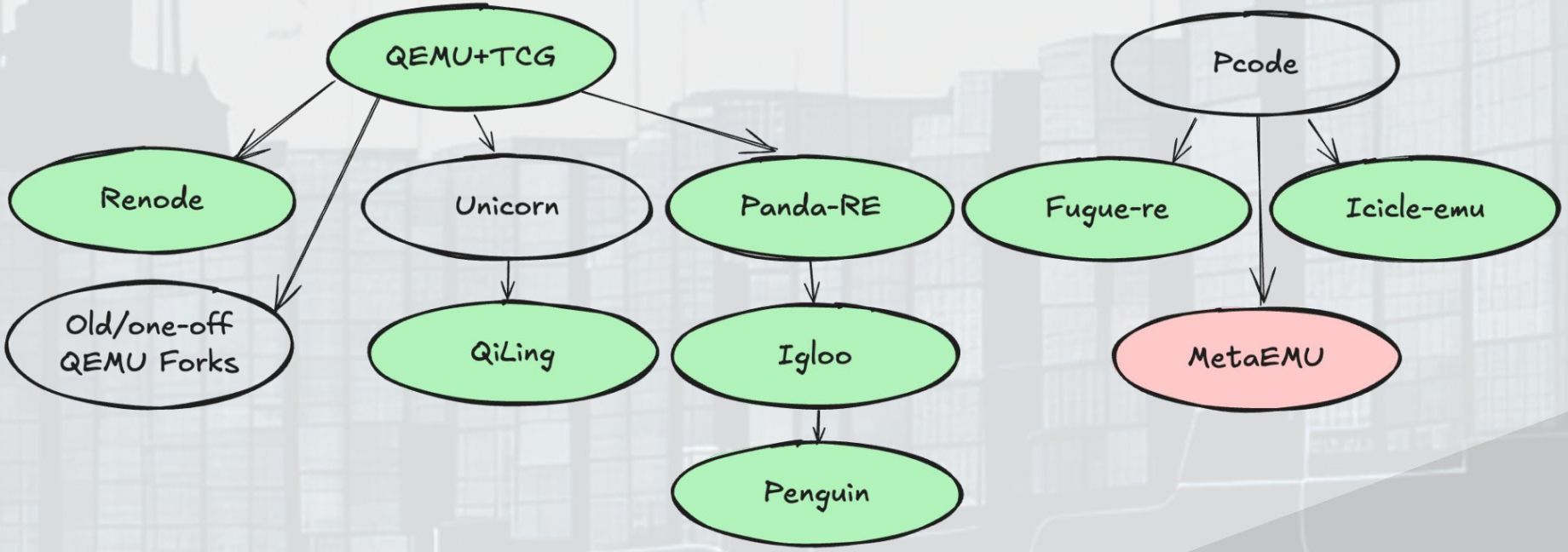


Open Source Emulator Family Tree



Unreleased

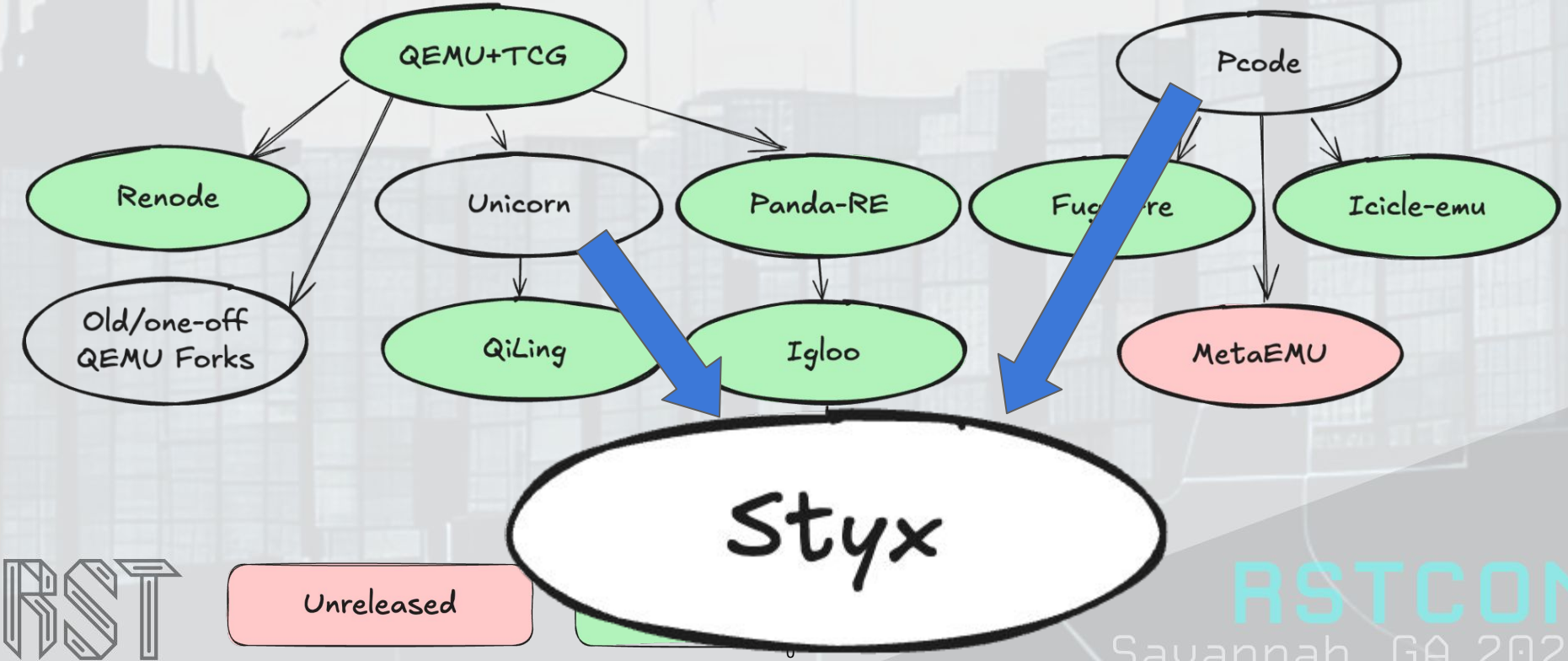
Open Source Emulator Family Tree



Unreleased

Linux/Linux User Focus

Open Source Emulator Family Tree



Challenges with Prior Solutions

- Not user configurable
- Licensing
- Challenging to support new ISA's
- Old forks of different tools
- Old forks of old forks of different tools
- Integration with other tools is "fun"
- abort();

Working With QEMU

- Adding things to QEMU is a pain
- QEMU is built to **run** software
- QEMU is *rigid*
- How do you model harvard memory in QEMU?
- No easy way to programmatically interface with QEMU I/O
- What if your SoC has multiple processors?

Old Forks of Tools

- “Full System Emulation” (Skips all hardware initialization)
- “Framework for oneoff target”
- Research contributions are “combining two incompatible QEMU forks”
- Some forks (panda-re, qiling etc.) get mild maintenance, locked on old QEMU
- 10yo QEMU checkout

abort();

qemu/CODING_STYLE.rst

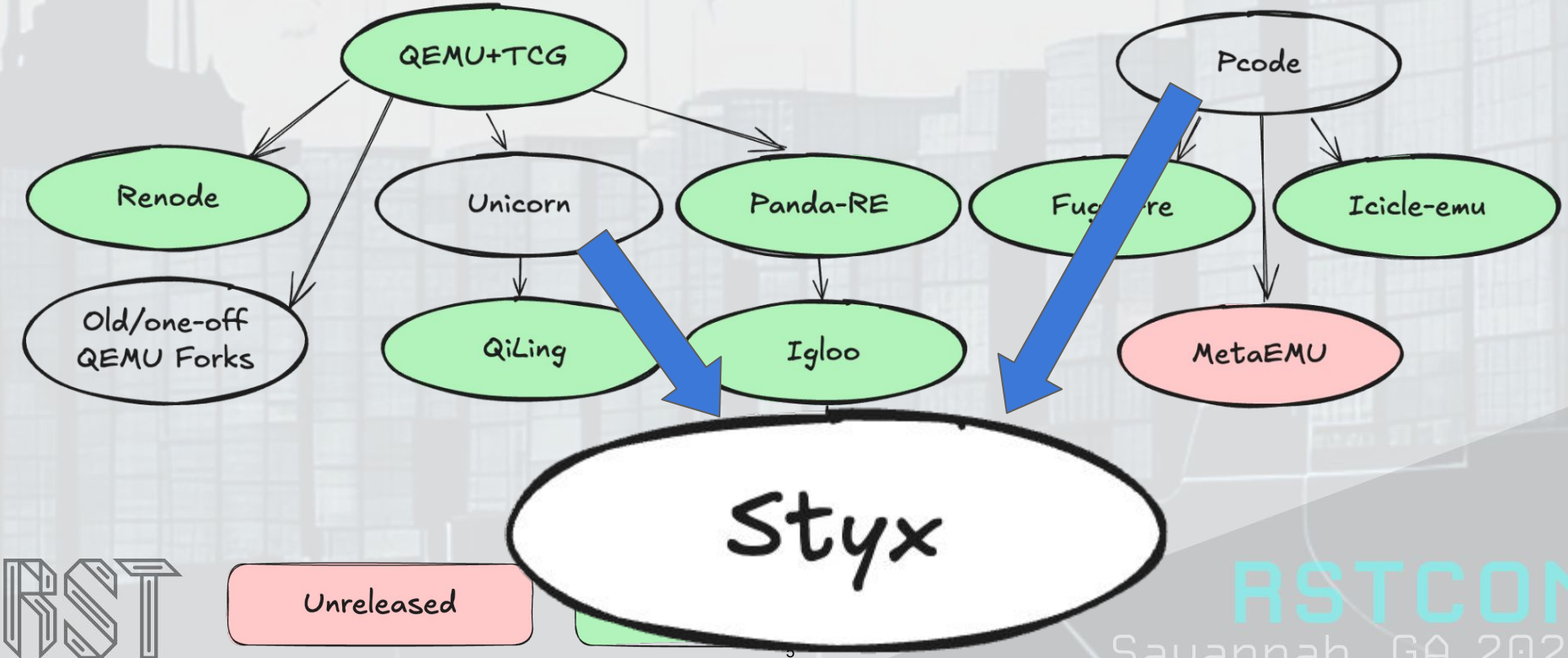
```
588
589  Do not call exit() or abort() to handle an error that can be triggered
590  by the guest (e.g., some unimplemented corner case in guest code
594  Note that &error_fatal is just another way to exit(1), and &error_abort
595  is just another way to abort().
596
```

qemu/exec.c

```
780  {
781      abort();
782  }
799      fprintf(stderr, "Bad ram offset %" PRIx64 "\n", (uint64_t)addr);
800      abort();
801
```

Show 2 more matches

Open Source Emulator Family Tree



What is the Styx Emulator

- Composable Emulator
- Library designed to be **TAILORED** to **YOUR** use case
- Purpose built for **DEBUGGING** and building **SYSTEM UNDERSTANDING**
- Publicly Released on 23 September, 2025
- A collection of tools and infrastructure to enable **RAPID** development of emulators
- A new open source collaboration project

The Vision (Revisited)

- A uniform way to programmatically emulate and model systems
- A single emulation and modeling API to build with
- A unification of tools that already exist so you don't lose the work already invested
- A flexible emulator framework built to be tailored by users, not just developers

Table of Contents

- whoami
- Where we're at (And when you'd want to use Styx)
- Community Feedback
- Current development
- Upcoming development
- Other fun ideas

whoami

- lockbox
- Army for a few years
- CTF for a few more
- Software Engineer out of spite
- Initial author/co-maintainer of Styx Emulator
- Someone who wants better debugging tools!

Day job is Emulation + Environments development at Zealot Labs

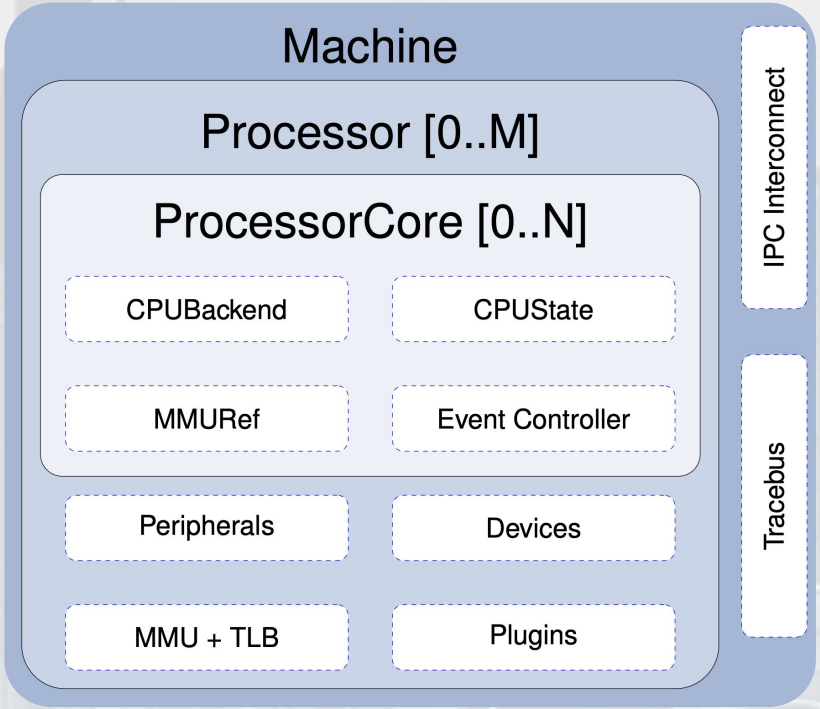
Styx: at the beginning

- Use UNICORN for common target set (ARM, PPC, etc.)
- Use PCODE interpreter for the uncommon targets (DSPs, CoProcessors)
- Build up initial Event Controller + Peripheral models
- Use gRPC for all network programming
- Create code generation tools wherever possible (from C SDK's, etc.)
- Initial focus was bug finding (now "debugging")

When to use the Styx Emulator

- When QEMU doesn't work / "Would take a long time in QEMU"
- Working with CoProcessors or DSPs
- When you need to **DEBUG** rather than run target programs
- Need multiple emulators to communicate
- Need programmatic control of peripherals
- You need to integrate emulation with other tools
- Fidelity and introspection is more important than speed

Where we're at



- Examples
- Extensive API Documentation
- Programmatic Peripherals
- Unified API
- Growing Target Support
- Unified Configuration
- Extensions and Plugins

Rust Minimal Example

```rust
// create a builder for the processor, this is a pretty bog
// standard builder-pattern that is common among rust projects
// due to the simple data-ownership
let builder = ProcessorBuilder::default()
    .with_builder(Kinetis21Builder::default())
    .with_loader(RawLoader)
    .add_plugin(ProcessorTracingPlugin)
    .with_target_program(get_firmware_path());

// "Build" the processor using the builder-pattern.
// All it does is consume all the inputs to create a final
// processor you can interact with and execute code with
let mut proc = builder.build()?;

// start the execution of the input `TargetProgram`
proc.run(Forever)?;
```

Python Minimal Example

```
builder = ProcessorBuilder()
builder.target_program = str(target_program_path())
builder.loader = RawLoader()
builder.executor = DefaultExecutor()
builder.add_plugin(ProcessorTracingPlugin())
proc = builder.build(Target.Kinetis21)

proc.start()
```

Embed In Python Integration Tests

```python
proc = build_processor(Backend.Pcode)
proc.add_hook(UnmappedFaultHook(0, 0xFFFFFFFF, unmapped_fault_hook))
proc.add_hook(ProtectionFaultHook(0, 0xFFFFFFFF, protection_fault_hook))

# unmapped address
proc.pc = 0x100004
proc.start(inst=10)
report = proc.wait_for_stop()
assert total["total"] == 1
assert report.instructions == 0
assert report.is_fatal
```

Rust Firmware/OS Integration Tests

```rust
let mut tests = Tests::default();
tests
    .add(Test::new_hook("Math", 0xfff0260c, &processor))
    .add(Test::new_hook("Coms (UART)", 0xfff02624, &processor))
    .add(Test::new_hook("Semaphore", 0xfff0263c, &processor))
    .add(Test::new_hook("Blocking Queues", 0xfff02654, &processor))
    .add(Test::new_hook(
        "Dynamic Priority Tasks",
        0xfff0266c,
        &processor,
    ))
    .add(Test::new_hook("Create Task", 0xfff02684, &processor))
    .add(Test::new_hook("Block Time", 0xfff0269c, &processor))
    .add(Test::new_hook("Generic Queues", 0xfff026b4, &processor))
    .add(Test::new_hook("Queue Peek", 0xfff026cc, &processor))
    .add(Test::new_hook("Counting Semaphore", 0xfff026e4, &processor))
    .add(Test::new_hook("Recursive Mutex", 0xfff026fc, &processor))
    .add(Test::new_hook_reg(
        "Reg Test Status",
        0xfff02718,
        &processor,
        Ppc32Register::R0,
        1,
    ))
```

Add New Target or ISA Support

- Documented support checklists:
  - New Target:
    - https://docs.styx-emulator.org/user/adding_a_processor.html
  - New ISA:
    - https://docs.styx-emulator.org/user/new_architectures.html
    - https://docs.styx-emulator.org/user/new_architectures_pcode.html

Keep a lookout for new blog post walkthroughs!

Current Target Support

- ISA's:
  - ARM32/64
  - MIPS32/64
  - PPC32
  - Blackfin
  - SuperH/2/3/4
  - Hexagon
- Peripherals:
  - UART
  - I2C
  - Ethernet
  - SPI
  - Timers
- Devices:
  - ADC
  - DAC
  - SPI Flash
  - SDMMC
  - RTC

Look in the repository / documentation site for specific pre-built processors

Built in Fuzzer

```rust
let mut proc = ProcessorBuilder::default()
    .with_builder(Kinetis21Builder::default())
    .with_backend(Backend::Unicorn)
    .with_executor(FuzzerExecutor::new(
        COVERAGE_MAP_SIZE,
        StyxFuzzerConfig {
            timeout: Duration::from_secs(1),
            branches_filepath: String::from("./branches.txt"),
            exits: vec![0xba2],
            max_input_len: MAX_INPUT_LEN,
            input_hook: Box::new(insert_input),
            setup: Box::new(pre_fuzzing_setup),
            context_restore: Box::new(context_restore),
            context_save: Box::new(context_save),
            ..Default::default()
        },
    ))
    .add_plugin(StyxTracePlugin::new(false, false, false, true))
    .with_loader(RawLoader)
    .with_target_program(get_firmware_path())
    .build()?;
```

Built in Fuzzer

```
cargo run --release --features tui

Styx-LibAFL Fuzzer (default)

charts
 speed | corpus | objectives (`g` switch)

process timing
run time            0h-1m-38s
exec speed          46.81k

speed chart
46819
0h-0m-0s                    0h-0m-45s          0h-1m-32s

client #0 (l/r arrows to switch)
process timing
run time                0h-0m-0s
exec speed              0
last new entry          0h-0m-0s
last solution           0h-0m-0s

item geometry
pending                 0
pend fav                0
own finds               0
imported                0
stability               0%

generic
corpus count            8
total execs             4286035
map density             0%

overall results
cycles done             0
solutions               13

clients logs (`t` to show/hide)
[UserStats   #0] corpus: 0, objectives: 0, executions: 0, exec/sec: 0.000, coverage map: 2/1024 (
[UserStats   #0] corpus: 0, objectives: 0, executions: 0, exec/sec: 0.000, coverage map: 2/1024 (
[Testcase    #0] corpus: 1, objectives: 0, executions: 1, exec/sec: 0.000, coverage map: 2/1024 (
[UserStats   #0] corpus: 1, objectives: 0, executions: 1, exec/sec: 0.000, coverage map: 3/1024 (
[UserStats   #0] corpus: 1, objectives: 0, executions: 1, exec/sec: 0.000, coverage map: 3/1024 (
[Testcase    #0] corpus: 2, objectives: 0, executions: 9, exec/sec: 0.000, coverage map: 3/1024 (
[UserStats   #0] corpus: 2, objectives: 0, executions: 9, exec/sec: 0.000, coverage map: 5/1024 (
[UserStats   #0] corpus: 2, objectives: 0, executions: 9, exec/sec: 0.000, coverage map: 5/1024 (
[Testcase    #0] corpus: 3, objectives: 0, executions: 10, exec/sec: 0.000, coverage map: 5/1024
```

Built in Debugger

```rust
// build the processor
let mut proc_builder = ProcessorBuilder::default()
    .with_builder(PowerPC405Builder::default())
    .with_executor(GdbExecutor::<Ppc4xxTargetDescription>::new(gdb_params)?)
    .with_loader(ParameterizedLoader::default()) // takes an input yaml
    .with_input_bytes(loader_yaml.as_bytes().into());

// actually "build" the processor now that all the options
// are initialized
let mut proc = proc_builder.build()?;

// start `TargetProgram` execution
proc.run(Forever)?;
```

Built in Debugger

```rust
// build the processor
let mut proc_builder = ProcessorBuilder::default()
    .with_executor(GdbExecutor::<Ppc4xxTargetDescription>::new(gdb_params)?)
    .with_input_bytes(loader_yaml.as_bytes().into());

// actually "build" the processor now that all the options
// are initialized
let mut proc = proc_builder.build()?;

// start `TargetProgram` execution
proc.run(Forever)?;
```

Interact with Packaged Devices

```
let eeprom = AT25HP512::new();
let adc = ADS7866::new();
let dac = RHRDAC121::new(None);

let client0 = SPIClient::new(args.to_string(), 0);
client0.connect_device(eeprom);

let client1 = SPIClient::new(args.to_string(), 1);
client1.connect_device(adc);

let client2 = SPIClient::new(args.to_string(), 2);
client2.connect_device(dac);
```

Multi-Emulator Tracebus

- Tracebus allows some cool things
- **Compile out** trace instrumentation you don't use
- **Interrupt** data Event
- **Concurrent** emulator support

```rust
#[derive(Clone, Debug)]
pub enum TraceableItem {
    BlockTraceEvent(TraceEventType::BLOCK),
    BranchEvent(TraceEventType::BRANCH),
    ControlEvent(TraceEventType::CTRL),
    InsnExecEvent(TraceEventType::INST_EXEC),
    InsnFetchEvent(TraceEventType::INST_FETCH),
    InterruptEvent(TraceEventType::INTERRUPT),
    MemReadEvent(TraceEventType::MEM_READ),
    MemWriteEvent(TraceEventType::MEM_WRT),
    RegReadEvent(TraceEventType::REG_READ),
    RegWriteEvent(TraceEventType::REG_WRITE),
}
```

External Tools Integration

- MVP Web UI to manage emulations and traces
- MVP Ghidra interop and data streaming
- Live tracebus visualizations
- Communicate with peripherals via gRPC (protobuf over the wire) enabling custom development in **any programming language**

Community Feedback

- Targets used (SuperH, ARM32)
- “this thing is amazing. Immediate automatic first choice for me”
- Why not renode?
- Make it easier to create new components
- Publish on package registries when?

Community (New Components)

Commit 694b1f5

Browse files

lockbox committed 3 weeks ago

```
feat: add templates for common traits
```

docs-update (styx-emulator/styx-emulator#113)

1 parent 356f067 commit 694b1f5

Filter files...

15 files changed

+1558 -0 lines changed

Search within co

Current Development

- Qualcomm + Android progress
- User Documentation
- Debugger frontend
- Unified Configuration

Current Development (Qualcomm)

- Recently merged
- Specific SoC support coming soon
- Peripherals and booting modem RTOS + Linux soon
- Longest ISA TTS “Time to Support” (3 months)
- Most complicated ISA to date

Hexagon ISA Execution Support ✓ 116

#104 by yuv418 was merged last week • Approved 5 tasks done

Current Development (Documentation)

- Merge Request in Progress
- Adds support for generating new components to expedite development
- Updates specific documentation

WIP: Docs update 1

#113 opened last month by lockbox • Draft 4 of 6 tasks

Current Development (Debugger)

- Unifying debug access via a single API
- Planning an integration with BinaryNinja debugger
- Still in the planning phase
- Using the in-tree protobuf definitions as a starting point

Current Development (Unified Configura

“One YAML to rule them all”

```yaml
version: 1
processors:
- name: FreeRTOS Processor
  processor: ppc_4xx
  backend: Pcode
  executor:
    id: gdb
    config:
      connection: 127.0.0.1:9999
      arch: Ppc405
      verbose: true
  program:
  - !FileRaw
    base: 0xfff00000
    file: ../../data/test-binaries/ppc/ppc405/bin/freertos.bin
    perms: !AllowAll
  - !RegisterImmediate
    register: pc
    value: 0xfffffffc
```

Upcoming Development

- Arm32 + Arm64 hypervisor support
- Unified interfaces for instruction execution backend
- Android phone emulators
- Tms320c2xxx
- Remote emulation orchestration
- Device-tree driven emulation (“automatic emulation” 😂)

Some Research Ideas

- Make trace analysis great again
- Time travel
- Generic support for cross-emulator time synchronization
- Fast multi-host architecture software TLB + MMU implementation
- Something better than SLEIGH (but lowers to SLEIGH for compatibility)
- Compile time tunable instruction decoding (via zig?)
- Emulation IR around webassembly
- Styx-simulation api to create a generic simulation API for physics simulators + plugins

Other Fun Ideas

- Styx-tcg backend
- Tms320c67xx
- Complete stm32 family support
- Complete AVR8/16/32 family support
- Vscode integration
- Zephyr + pigweed integration
- Support quadcopter development projects with emulators (eg. betaflight etc.)
- Styx-machines library of reusable machines for people
- Formal gnuradio blocks + plugin for Styx as a sink / source

Summary

- The Styx Emulator is a new **USER** configurable emulator framework
- Easy to extend
- Easy to integrate with external tools
- Easy to customize for **YOUR** use case
- Focusing on Embedded and DSP platforms

Links

Official Repository: https://github.com/styx-emulator
Official Twitter/X: https://x.com/styx_emulator
Official Mastodon: https://infosec.exchange/@styx_emulator
Official Docs site: https://docs.styx-emulator.org
Community Discord: https://discord.gg/styx-emulator

Personal:

Email: lockbox@struct.foo
Github: https://github.com/lockbox
Blog: https://stumbl.ing

Slides:


RST
RSTCON
Savannah, GA 2025