2021.NET Conf China

开源共建 | 开放创新 | 开发赋能

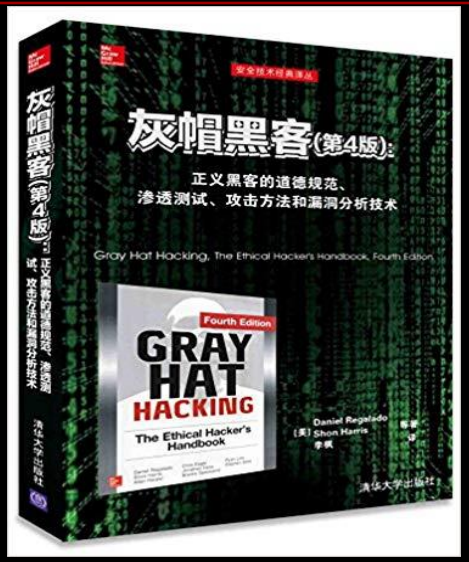
Renode: a Swiss Army Knife for RISC-V development

Feng Li (李枫)
hkli2013@126.com

2021 中国 .NET 开发者大会

Who Am I

- The main translator of the book «Gray Hat Hacking The Ethical Hacker's Handbook, Fourth Edition» (ISBN：9787302428671) & «Linux Hardening in Hostile Networks, First Edition» (ISBN: 9787115544384)



- Pure software developmement for ~15 years
- Actively participate in various activities of the open source community
- https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/Conf
  https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/LTS
- Recently, focus on infrastructure of Cloud/Edge Computing, AI, Virtualization, Program Runtimes, Network, 5G, RISC-V, EDA…

Agenda

I. Tech Stack

- System Modeling and Simulation
- HW/SW Co-design
- HW/SW FOSS Tools
- RISC-V
- IoT Edge
- .Net
- Testbed

II. What is Renode

- Overview
- Architecture & Design

III. Renode for RISC-V development

- Built-in Support
- Getting Started
- Demos
- Co-simulation

- Debugging
- Parallel HW/SW Development

IV. Renode for Edge AI

- Overview
- VEDLIoT
- Renode can help

V. Renode on RPi4B

- Porting
- A special prebuilt version
- Virtualization
- Renode on GraalVM

VI. Wrap-up

I. Tech Stack

2021.NET Conf China
开源共建 | 开放创新 | 开发赋能

1) System Modeling and Simulation

- https://en.wikipedia.org/wiki/Modeling_and_simulation

Modeling and simulation (**M&S**) is the use of models (e.g., physical, mathematical, or logical representation of a system, entity, phenomenon, or process) as a basis for simulations to develop data utilized for managerial or technical decision making.[1][2]

In the computer application of modeling and simulation a computer is used to build a mathematical model which contains key parameters of the physical model. The mathematical model represents the physical model in virtual form, and conditions are applied that set up the experiment of interest. The simulation starts – i.e., the computer calculates the results of those conditions on the mathematical model – and outputs results in a format that is either machine- or human-readable, depending upon the implementation.

The use of M&S within engineering is well recognized. Simulation technology belongs to the tool set of engineers of all application domains and has been included in the body of knowledge of engineering management. M&S helps to reduce costs, increase the quality of products and systems, and document and archive lessons learned. Because the results of a simulation are only as good as the underlying model(s), engineers, operators, and analysts must pay particular attention to its construction. To ensure that the results of the simulation are applicable to the real world, the user must understand the assumptions, conceptualizations, and constraints of its implementation. Additionally, models may be updated and improved using results of actual experiments. M&S is a discipline on its own. Its many application domains often lead to the assumption that M&S is a pure application. This is not the case and needs to be recognized by engineering management in the application of M&S.

The use of such mathematical models and simulations avoids actual experimentation, which can be costly and time-consuming. Instead, mathematical knowledge and computational power is used to solve real-world problems cheaply and in a time efficient manner. As such, M&S can facilitate understanding a system's behavior without actually testing the system in the real world. For example, to determine which type of spoiler would improve traction the most while designing a race car, a computer simulation of the car could be used to estimate the effect of different spoiler shapes on the coefficient of friction in a turn. Useful insights about different decisions in the design could be gleaned without actually building the car. In addition, simulation can support experimentation that occurs totally in software, or in human-in-the-loop environments where simulation represents systems or generates data needed to meet experiment objectives. Furthermore, simulation can be used to train persons using a virtual environment that would otherwise be difficult or expensive to produce.

Interest in simulations

-

Interest in simulations [edit]

Technically, simulation is well accepted. The 2006 National Science Foundation (NSF) Report on "Simulation-based Engineering Science"[3] showed the potential of using simulation technology and methods to revolutionize the engineering science. Among the reasons for the steadily increasing interest in simulation applications are the following:

1. Using simulations is generally cheaper, safer and sometimes more ethical than conducting real-world experiments. For example, supercomputers are sometimes used to simulate the detonation of nuclear devices and their effects in order to support better preparedness in the event of a nuclear explosion.[4] Similar efforts are conducted to simulate hurricanes and other natural catastrophes.[5][6][7]
   See also: Hurricane Weather Research and Forecasting model
2. Simulations can often be even more realistic than traditional experiments, as they allow the free configuration of environment parameters found in the operational application field of the final product. Examples are supporting deep water operation of the US Navy or the simulating the surface of neighbored planets in preparation of NASA missions.
3. Simulations can often be conducted faster than real time. This allows using them for efficient if-then-else analyses of different alternatives, in particular when the necessary data to initialize the simulation can easily be obtained from operational data. This use of simulation adds decision support simulation systems to the tool box of traditional decision support systems.
4. Simulations allow setting up a coherent synthetic environment that allows for integration of simulated systems in the early analysis phase via mixed virtual systems with first prototypical components to a virtual test environment for the final system. If managed correctly, the environment can be migrated from the development and test domain to the training and education domain in follow-on life cycle phases for the systems (including the option to train and optimize a virtual twin of the real system under realistic constraints even before first components are being built).

The military and defense domain, in particular within the United States, has been the main M&S champion, in form of funding as well as application of M&S. E.g., M&S in modern military organizations is part of the acquisition/procurement strategy. Specifically, M&S is used to conduct Events and Experiments that influence requirements and training for military systems. As such, M&S is considered an integral part of systems engineering of military systems. Other application domains, however, are currently catching up. M&S in the fields of medicine, transportation, and other industries is poised to rapidly outstrip DoD's use of M&S in the years ahead, if it hasn't already happened.[8]

Key Concepts

- https://en.wikipedia.org/wiki/Mathematical_model
- https://en.wikipedia.org/wiki/Computer_simulation
- https://en.wikipedia.org/wiki/Simulation
- …

Simulation vs Emulation

- https://www.perfecto.io/blog/emulation-vs-simulation
- https://www.geeksforgeeks.org/difference-between-emulation-and-simulation/
- https://en.wikipedia.org/wiki/Emulator
- …

2) HW/SW Co-design

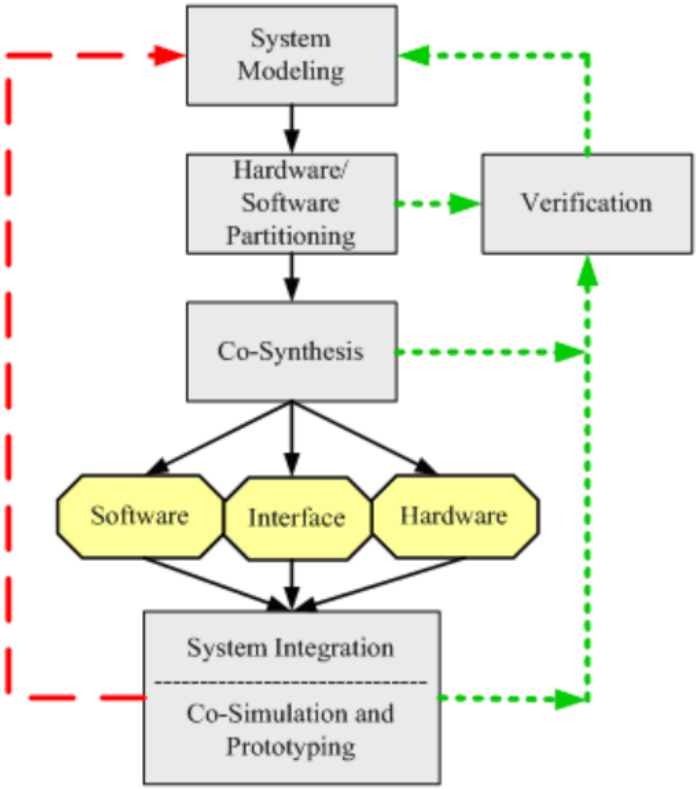
University of Victoria

- https://webhome.cs.uvic.ca/~mserra/HScodesign.html
- The co-design of HW/SW systems may be viewed as composed of four main phases as illustrated in in the side diagram:

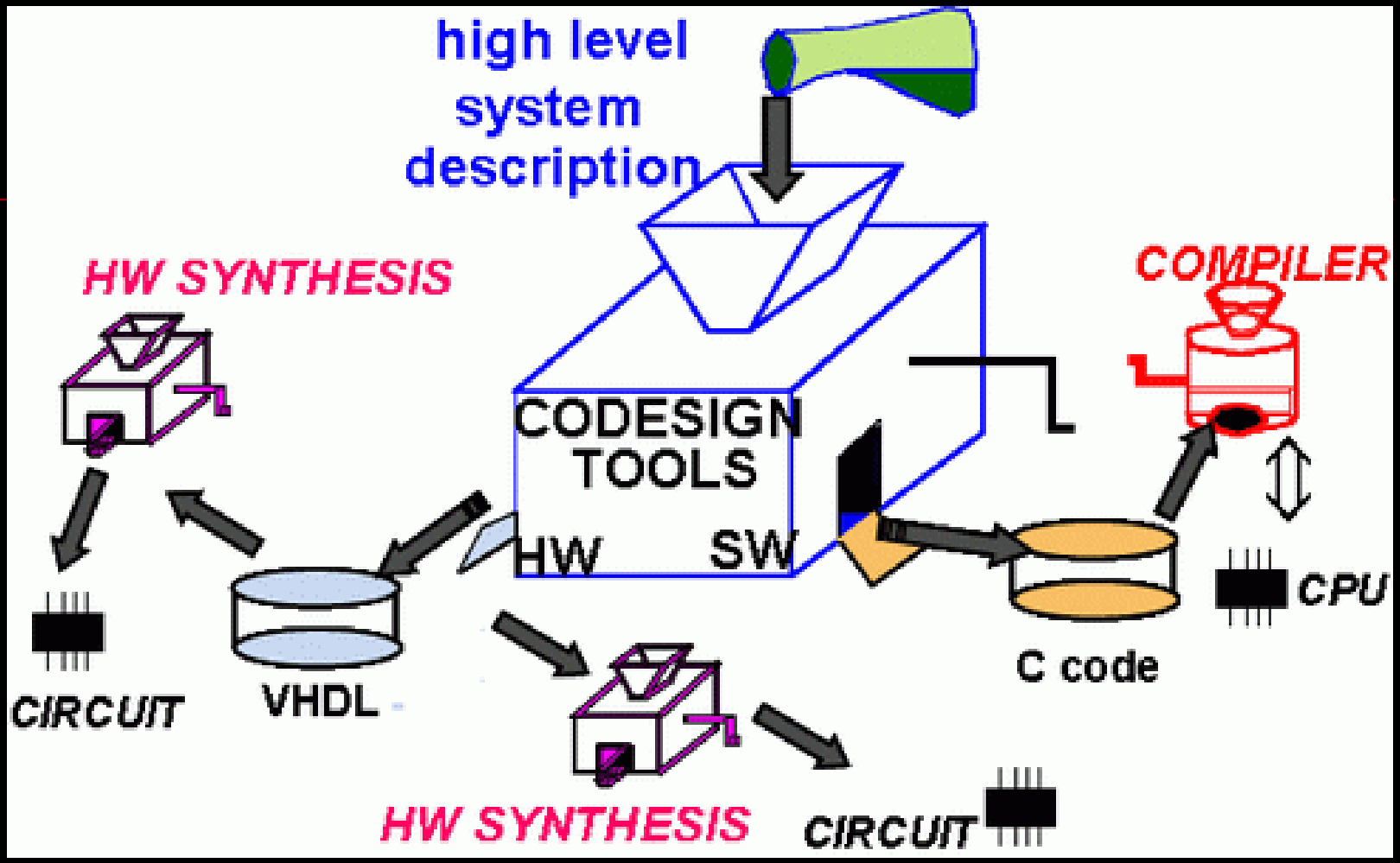
  1. Modeling
  2. Partitioning
  3. Co-Synthesis
  4. C-Simulation

  Modeling involves specifying the concepts and the constraints of the system to obtain a refined specification. This phase of the design also specifies a software and hardware model. The first problem is to find a suitable specification methodology for a target system. Some researchers favour a formal language which can yield provably-correct code. But most prefer a hardware-type language (e.g., VHDL, Verilog), a software-type language (C, C++, Handel-C, SystemC), or other formalism lacking a hardware or software bias (such as Codesign Finite State Machines). There are three different paths the modeling process can take, considering its starting point:

  - An existing software implementation of the problem.
  - An existing hardware, for example a chip, is present.
  - None of the above is given, only specifications leaving an open choice for a model.



2021.NET Conf China
开源共建 | 开放创新 | 开发赋能



2.1 Evolution of HDL

- https://en.wikipedia.org/wiki/Hardware_description_language
- Verilog/VHDL → HLS → eDSL

SystemVerilog, SystemC...

High-level Synthesis
https://en.wikipedia.org/wiki/High-level_synthesis

embedded Domain Specific Language

Typical eDSLs

- Haskell as host
  Bluespec, Clash...
- Scala as host
  Chisel, SpinalHDL...
- Python as host
  FHDL...
- Finally convert to Verilog or VHDL
  https://en.wikipedia.org/wiki/Source-to-source_compiler

2.2 ASIP

- https://en.wikipedia.org/wiki/Application-specific_instruction_set_processor

An **application-specific instruction set processor** (**ASIP**) is a component used in system-on-a-chip design. The instruction set of an ASIP is tailored to benefit a specific application. This specialization of the core provides a tradeoff between the flexibility of a general purpose CPU and the performance of an ASIC.

Some ASIPs have a configurable instruction set. Usually, these cores are divided into two parts: *static* logic which defines a minimum ISA (instruction-set architecture) and *configurable* logic which can be used to design new instructions. The configurable logic can be programmed either in the field in a similar fashion to an field-programmable gate array (FPGA) or during the chip synthesis.

ASIPs can be used as an alternative of hardware accelerators for baseband signal processing[1] or video coding.[2] Traditional hardware accelerators for these applications suffer from inflexibility. It is very difficult to reuse the hardware datapath with handwritten finite-state machines (FSM). The retargetable compilers of ASIPs help the designer to update the program and reuse the datapath. Typically, the ASIP design is more or less dependent on the tool flow because designing a processor from scratch can be very complicated. One approach is to describe the processor using a high level language and then to automatically generate the ASIP's software toolset.[3] There are some commercial tools to design ASIPs from a high-level language, for example ASIP Designer from Synopsys or Studio from Codasip. There is an open source tool as well, TTA-based co-design environment (TCE).

3) HW/SW FOSS Tools

3.1 QEMU

- https://en.wikipedia.org/wiki/QEMU

QEMU (short for **Q**uick **EMU**lator[2][*dubious – discuss*]) is a free and open-source emulator and virtualizer that can perform hardware virtualization.

QEMU is a hosted virtual machine monitor: it emulates the machine's processor through dynamic binary translation and provides a set of different hardware and device models for the machine, enabling it to run a variety of guest operating systems. It also can be used with Kernel-based Virtual Machine (KVM) to run virtual machines at near-native speed (by taking advantage of hardware extensions such as Intel VT-x). QEMU can also do emulation for user-level processes, allowing applications compiled for one architecture to run on another.[3]

Operating modes

User-mode emulation

In this mode QEMU runs single Linux or Darwin/macOS programs that were compiled for a different instruction set. System calls are thunked for endianness and for 32/64 bit mismatches. Fast cross-compilation and cross-debugging are the main targets for user-mode emulation.

System emulation

In this mode QEMU emulates a full computer system, including peripherals. It can be used to provide virtual hosting of several virtual computers on a single computer. QEMU can boot many guest operating systems, including Linux, Solaris, Microsoft Windows, DOS, and BSD;[6] it supports emulating several instruction sets, including x86, MIPS, 32-bit ARMv7, ARMv8, PowerPC, SPARC, ETRAX CRIS and MicroBlaze.

KVM Hosting

Here QEMU deals with the setting up and migration of KVM images. It is still involved in the emulation of hardware, but the execution of the guest is done by KVM as requested by QEMU.

Xen Hosting

QEMU is involved only in the emulation of hardware; the execution of the guest is done within Xen and is totally hidden from QEMU.

TCG(Tiny Code Generator)

The Tiny Code Generator (TCG) aims to remove the shortcoming of relying on a particular version of GCC or any compiler, instead incorporating the compiler (code generator) into other tasks performed by QEMU at run time. The whole translation task thus consists of two parts: basic blocks of target code (*TBs*) being rewritten in *TCG ops* - a kind of machine-independent intermediate notation, and subsequently this notation being compiled for the host's architecture by TCG. Optional optimisation passes are performed between them, for a just-in-time compiler (JIT) mode.

TCG requires dedicated code written to support every architecture it runs on, so that the JIT knows what to translate the *TCG ops* to. If no dedicated JIT code is available for the architecture, TCG falls back to a slow interpreter mode called TCG Interpreter (TCI). It also requires updating the target code to use TCG ops instead of the old *dyngen* ops.

- https://www.qemu.org/

A generic and open source machine emulator and virtualizer.

https://github.com/qemu/qemu

QEMU is capable of emulating a complete machine in software without any need for hardware virtualization support. By using dynamic translation, it achieves very good performance. QEMU can also integrate with the Xen and KVM hypervisors to provide emulated hardware while allowing the hypervisor to manage the CPU. With hypervisor support, QEMU can achieve near native performance for CPUs. When QEMU emulates CPUs directly it is capable of running operating systems made for one machine (e.g. an ARMv7 board) on a different machine (e.g. an x86_64 PC board).

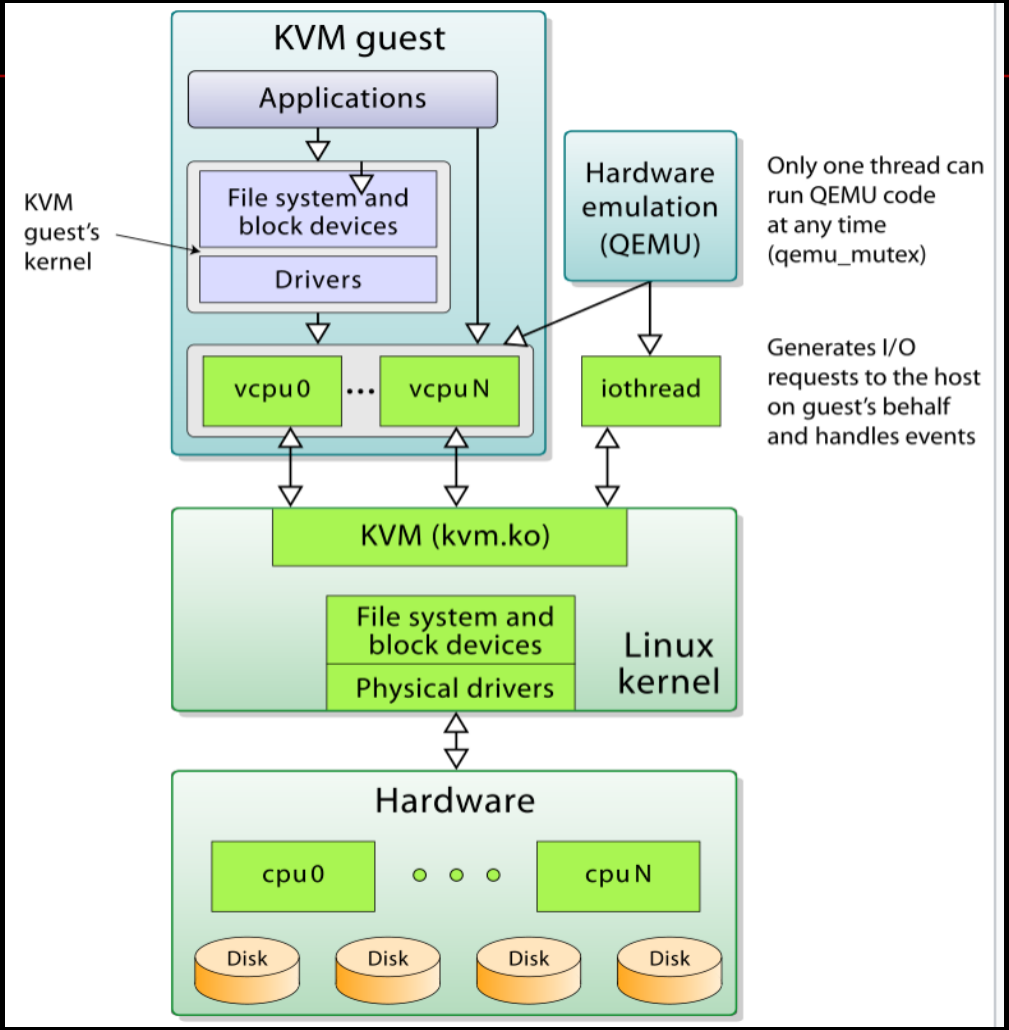
QEMU is also capable of providing userspace API virtualization for Linux and BSD kernel interfaces. This allows binaries compiled against one architecture ABI (e.g. the Linux PPC64 ABI) to be run on a host using a different architecture ABI (e.g. the Linux x86_64 ABI). This does not involve any hardware emulation, simply CPU and syscall emulation.

QEMU aims to fit into a variety of use cases. It can be invoked directly by users wishing to have full control over its behaviour and settings. It also aims to facilitate integration into higher level management layers, by providing a stable command line interface and monitor API. It is commonly invoked indirectly via the libvirt library when using open source applications such as oVirt, OpenStack and virt-manager.

QEMU as a whole is released under the GNU General Public License, version 2. For full licensing details, consult the LICENSE file.

- https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine

A high-level overview of the KVM/QEMU virtualization environment



- https://www.accellera.org/news/events/systemc-evolution-fika/systemc-evolution-fika-june-2021

Extending Qemu for EDA virtualization

- MULTI Thread Qemu
  - A massive speed improvement for Qemu to take advantage of multi-core hosts

- SystemC integration
  - The ability to mix SystemC models with Qemu.

- Instruction counting and analysis features
  - Enhanced counting mechanisms for memory accesses and instruction counting

Very approximatly, QEMU executes 15 host instructions for 1 guest instruction

accellera SYSTEMS INITIATIVE

SYSTEMC EVOLUTION FIKA

3.1.1 GreenSoC

- https://www.greensocs.com/

QEMU and SystemC Virtual Platforms.

Real Time QEMU/SystemC
Syncronise your watches - we run in real time!

Fast As Possible: HW In The Loop (HIL)
Run your models as fast as they can, to keep up with HIL!

DEBUG Mode
Find those impossible threading issues with pseudo-random deterministic mode.

Deterministic Mode
Run the model the same way every time.

Quantum Time
Quantum windowing, and TLM compliance.

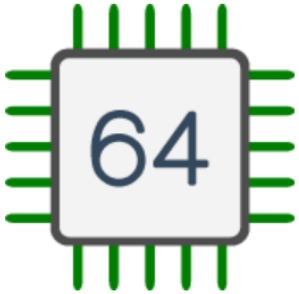
Non-Deterministic reverse debug
Replay what just happened, jump to the bug.

Save time and money

Let's do it! You will design better devices, better tuned to the software needs, and you will build them quicker.

Not only will you save time and money by adopting model based virtual platforms, but at GreenSocs you get our time and expertise.

No extra license cost !



Virtual Prototyping

- ✔ Multi-core and multi-thread solutions.
- ✔ Full Eclipse environment,
- ✔ Debug of both hardware model and software guest (through a GDB Stub).
- ✔ Our code is SystemC standards compliant and works with any tool, e.g. hardware synthesis.
- ✔ Extensive debug, verification and configuration features

Secure your design or IP

Virtual Platforms will bring your software development forward. Explore, design and optimise your architectures.

open source initiative

Open Source Forever

GreenSocs encourages open source development providing free and open source solutions based on SystemC, QEMU and GEM5. All open source products are free to use forever .

Your virtual platform free to use forever

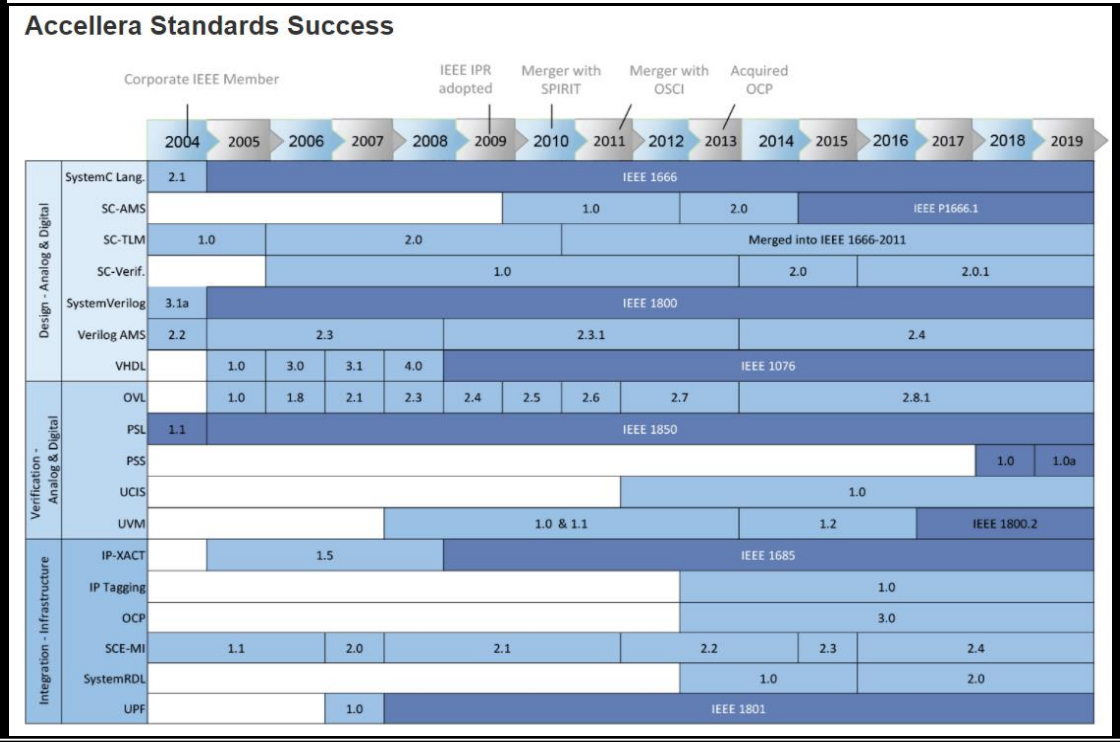
- **https://github.com/GreenSocs**

Accellera/SystemC

- https://www.accellera.org/

Who We Are

Accellera Systems Initiative is an independent, not-for profit organization dedicated to create, support, promote, and advance system-level design, modeling, and verification standards for use by the worldwide electronics industry. We are composed of a broad range of members that fully support the work of our technical committee to develop technology standards that are balanced, open, and benefit the worldwide electronics industry. Leading companies and semiconductor manufacturers around the world are using our electronic design automation (EDA) and intellectual property (IP) standards in a wide range of projects in numerous application areas to develop consumer, mobile, wireless, automotive, and other "smart" electronic devices. Through an ongoing partnership with the IEEE, standards and technical implementations developed by Accellera Systems Initiative are contributed to the IEEE for formal standardization and ongoing governance.



- https://en.wikipedia.org/wiki/SystemC

SystemC is a set of C++ classes and macros which provide an event-driven simulation interface (see also discrete event simulation). These facilities enable a designer to *simulate* concurrent processes, each described using plain C++ syntax. SystemC processes can communicate in a *simulated* real-time environment, using signals of all the datatypes offered by C++, some additional ones offered by the SystemC library, as well as user defined. In certain respects, SystemC deliberately mimics the hardware description languages VHDL and Verilog, but is more aptly described as a *system-level modeling language*.

SystemC is applied to system-level modeling, architectural exploration, performance modeling, software development, functional verification, and high-level synthesis. SystemC is often associated with electronic system-level (ESL) design, and with transaction-level modeling (TLM).

- https://en.wikipedia.org/wiki/Transaction-level_modeling

Transaction-level modeling (**TLM**) is a high-level approach to modeling digital systems where details of communication among modules are separated from the details of the implementation of functional units or of the communication architecture. Communication mechanisms such as buses or FIFOs are modeled as channels, and are presented to modules using SystemC interface classes. Transaction requests take place by calling interface functions of these channel models, which encapsulate low-level details of the information exchange. At the transaction level, the emphasis is more on the functionality of the data transfers – what data are transferred to and from what locations – and less on their actual implementation, that is, on the actual protocol used for data transfer. This approach makes it easier for the system-level designer to experiment, for example, with different bus architectures (all supporting a common abstract interface) without having to recode models that interact with any of the buses, provided these models interact with the bus through the common interface.[1]

However, the application of transaction-level modeling is not specific to the SystemC language and can be used with other languages. The concept of TLM first appears in system level language and modeling domain.[2]

- https://en.wikipedia.org/wiki/Electronic_system-level_design_and_verification

Electronic system level (ESL) design and verification is an electronic design methodology, focused on higher abstraction level concerns. The term *Electronic System Level* or *ESL Design* was first defined by Gartner Dataquest, an EDA-industry-analysis firm, on February 1, 2001.[1] It is defined in *ESL Design and Verification* [2] as: "the utilization of appropriate abstractions in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner."

The basic premise is to model the behavior of the entire system using a high-level language such as C, C++, or using graphical "model-based" design tools. Newer languages are emerging that enable the creation of a model at a higher level of abstraction including general purpose system design languages like SysML as well as those that are specific to embedded system design like SMDL and SSDL. Rapid and correct-by-construction implementation of the system can be automated using EDA tools such as high-level synthesis and embedded software tools, although much of it is performed manually today. ESL can also be accomplished through the use of SystemC as an abstract modeling language.

ESL is an established approach at many of the world' s leading System-on-a-chip (SoC) design companies, and is being used increasingly in system design.[3] From its genesis as an algorithm modeling methodology with 'no links to implementation', ESL is evolving into a set of complementary methodologies that enable embedded system design, verification, and debugging through to the hardware and software implementation of custom SoC, system-on-FPGA, system-on board, and entire multi-board systems.

Design and verification are two distinct disciplines within this methodology. Some practices are to keep the two elements separate, while others advocate for closer integration between design and verification.[4]

NVDLA virtual platform

- http://nvdla.org/vp.html

The NVDLA virtual platform provides a register-accurate system on which software can be quickly developed and debugged. The virtual platform is based on GreenSocs QBOX, a solution for co-simulation with QEMU and SystemC. Fig. 89 below shows the top level diagram of the NVDLA virtual simulator. A QEMU emulator of ARMv8 'virt' SoC board is included to provide high performance CPU emulation and generic devices. The emulator is wrapped inside a standard SystemC module with a set of TLM-2.0 interfaces to interact with other SystemC modules.

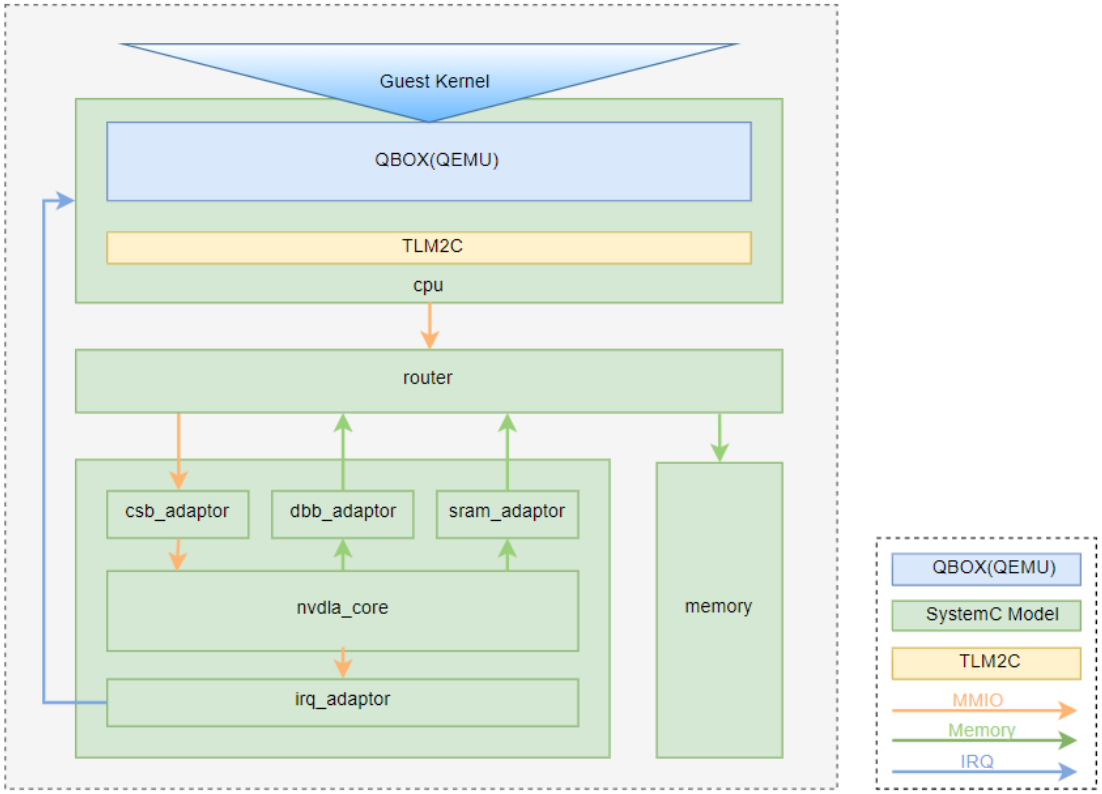


Fig. 89 - NVDLA Virtual Platform.

- https://github.com/nvdla/vp

4) RISC-V

4.1 Overview

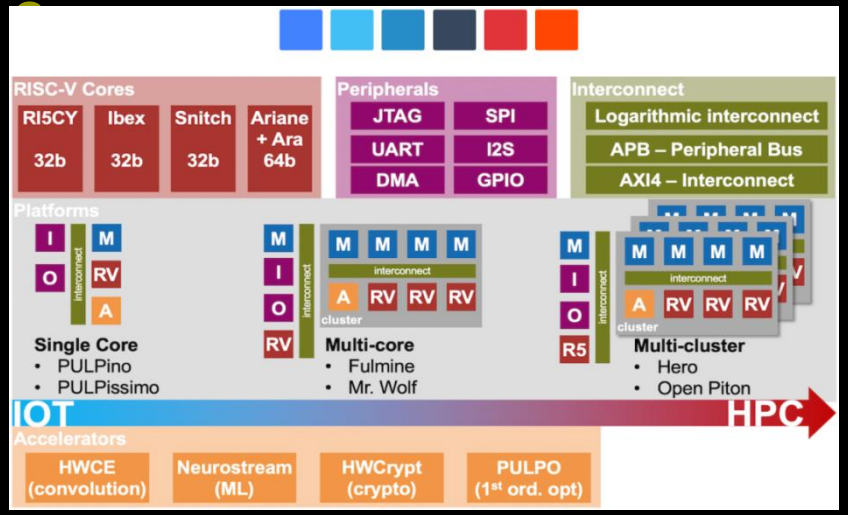
- https://en.wikipedia.org/wiki/RISC-V
- https://riscv.org/

Development

- https://riscv.org/exchange/
- https://github.com/riscv/riscv-isa-manual

Cores & SoC

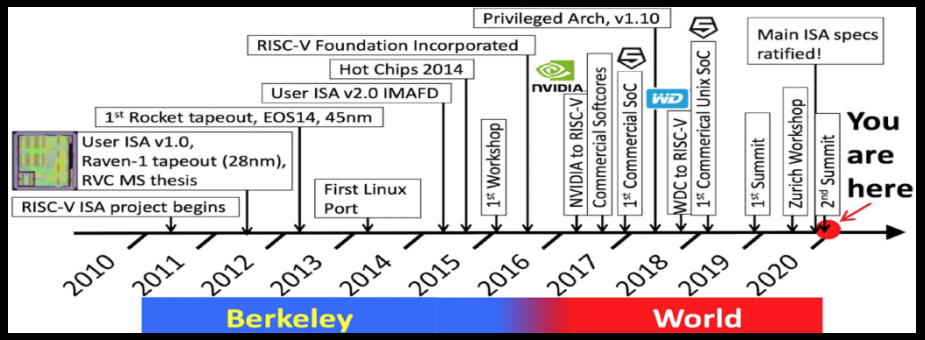
- https://github.com/riscv/riscv-cores-list
- https://riscv.org/exchange/cores-socs/
-



Source: https://fuentitech.com/what-is-the-risc-v-ecosystem/19227/

Ecosystem

- https://riscv.org/exchange/
- https://community.riscv.org/events/details/risc-v-foundation-bay-area-risc-v-group-presents-2021-risc-v-ecosystem-updates/

- 

Source: "Linux on RISC-V", D. Fustimi, ELC2020

- **Incredible industry progress**

- The European Processor Initiative finalized the first version of its **RISC-V accelerator architecture** and will deliver test chip in 2021.
- The RIOS Lab announced PicoRio, an affordable **RISC-V open source small-board computer** available in 2021.
- Imperas announced first **RISC-V verification reference model with UVM encapsulation**.
- Seagate announced **hard disk drive controller** with high-performance RISC-V CPU.
- GreenWaves **ultra-low power GAP9 hearables platform** enabling scene-aware and neural network-based noise reduction.
- Alibaba unveiled RV64GCV core in its Xuantie 910 processor for **cloud and edge servers**.
- Microchip released the first **SoC FPGA development kit** based on the RISC-V ISA.
- Andes released **superscalar multicore and L2 cache controller** processors.
- StarFive released the world's first **RISC-V AI visual processing platform**
- SiFive unveiled world's fastest development board for **RISC-V Personal Computers**.
- Micro Magic announced an incredibly **fast 64-bit RISC-V core** achieving 5GHz and 13,000 CoreMarks at 1.1V.

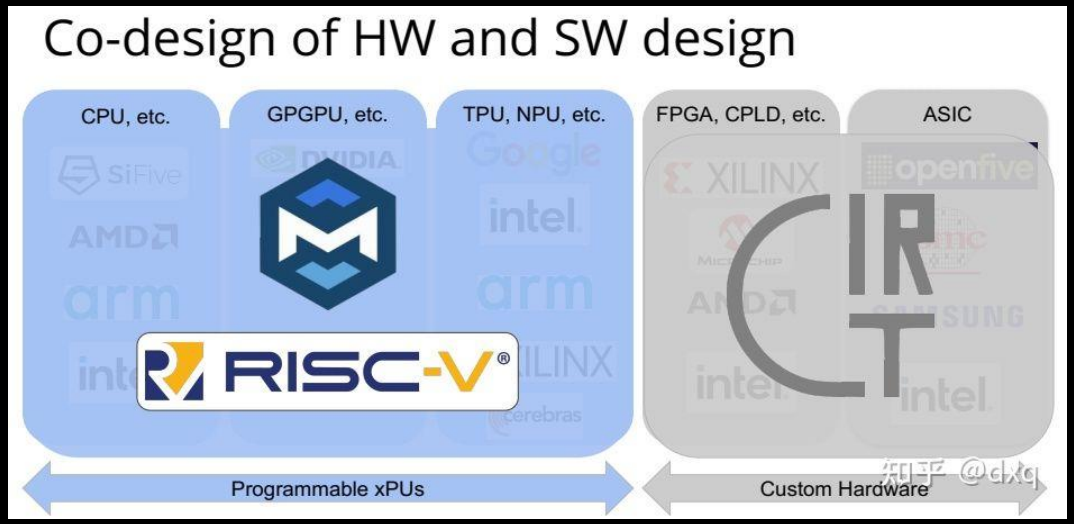
Source: "RISC-V: The Open Era of Computing", Calista Redmond, The Linux Foundation Spring Member Meeting 2021

4.2 A New Golden Age for Computer Architecture

- https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext

The Golden Age of Compiler Design in an Era of HW/SW Co-design

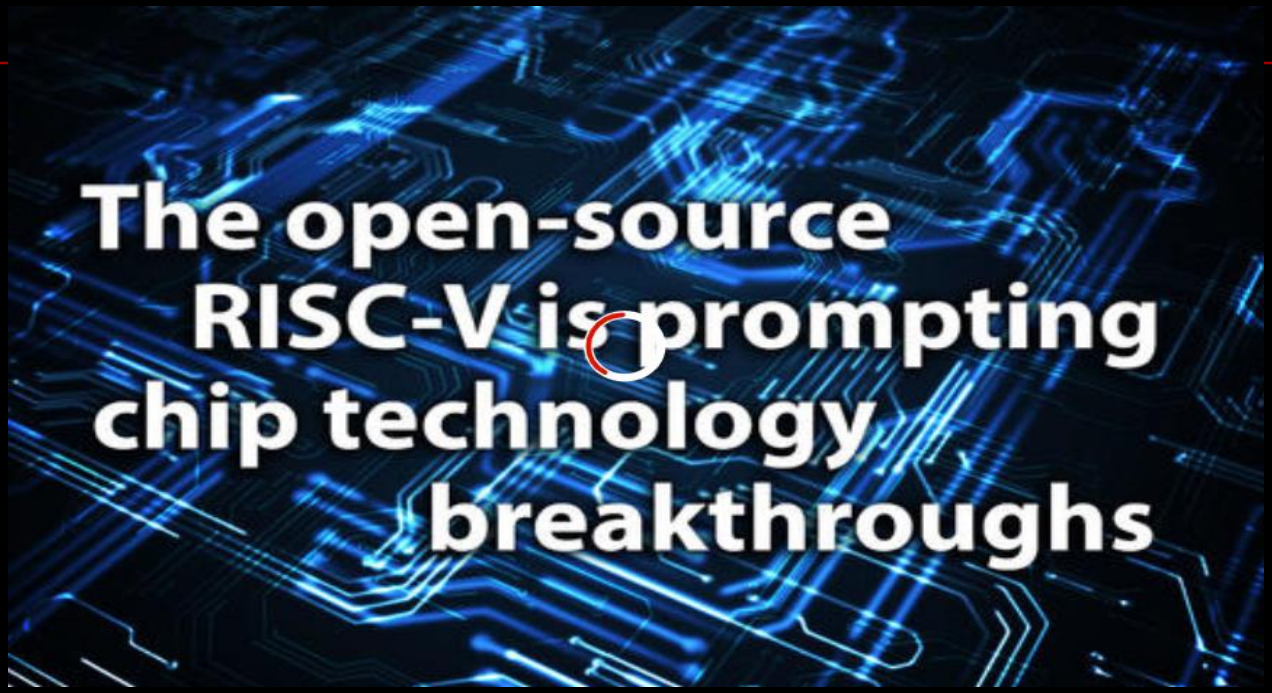
-



- **https://zhuanlan.zhihu.com/p/367035973 //看Chris Lattner在ASPLOS演讲有感**

“the Linux of the chip world”

- https://www.zdnet.com/article/risc-v-the-linux-of-the-chip-world-is-starting-to-produce-technological-breakthroughs/



4.2 Core/SoC Frameworks

4.2.1 LiteX

- http://www.enjoy-digital.fr/
- Based on Migen (Python for FPGA), LiteX SoC builder and the LiteX cores ecosystem allow us (and others :)) to create full modular/scalable FPGA based systems easily! Just give it a try!:

```
                    __   __
                   / /  (_) /____ | |/_/
                  / /__/ / __/ -_)>  <
                 /____/_/\__/\__/_/|_|
                       Migen inside

            Build your hardware, easily!
          Copyright 2012-2018 / EnjoyDigital

                   +---------------+
                   |FPGA toolchains|
                   +----^-----+----+
                        |     |
                      +--+-----V--+
          +-------+   |          |
          | Migen +-------->     |
          +-------+   |          |        Your design
                      |   LiteX  +---> ready to be used!
                      |          |
+----------------------+ |       |
|LiteX Cores Ecosystem +-->       |
+----------------------+  +-^-------^-+
 (Eth, SATA, DRAM, USB,     |       |
  PCIe, Video, etc...)      +       +
                          board   target
                          file    file
```

LiteX already supports various softcores CPUs: LM32, Mor1kx, PicoRV32, VexRiscv and is compatible with the LiteX's Cores Ecosystem:

- LiteDRAM: https://github.com/enjoy-digital/litedram
- LiteEth: https://github.com/enjoy-digital/liteeth
- LitePCIe: https://github.com/enjoy-digital/litepcie
- LiteSATA: https://github.com/enjoy-digital/litesata
- LiteUSB: https://github.com/enjoy-digital/liteusb
- LiteSDCard: https://github.com/enjoy-digital/litesdcard
- LiteICLink: https://github.com/enjoy-digital/liteiclink
- LiteJESD204B: https://github.com/enjoy-digital/litejesd204b
- LiteVideo: https://github.com/enjoy-digital/litevideo
- LiteScope: https://github.com/enjoy-digital/litescope

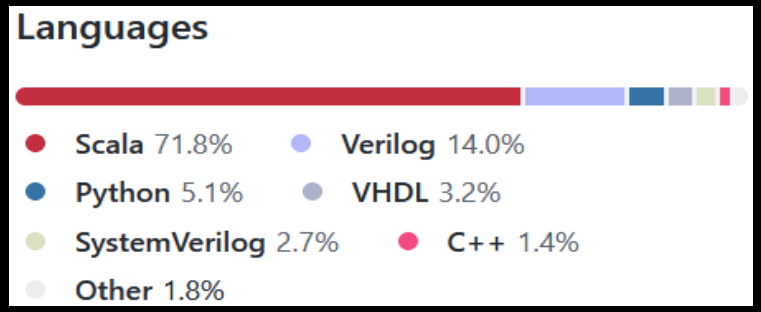
LiteX by EnjoyDigital
Build your hardware, easily!

- https://github.com/enjoy-digital
- Similar to Rocket Chip Generator, while mainly base on Python.
- For details, you may refer to my previous talk "Python-based Open Source Toolchain for RISC-V Development" at the 1st RISC-V World Conference China 2021(Shanghai) and the upcoming follow-ups.

4.2.2 SpinalHDL

- https://github.com/SpinalHDL/SpinalHDL

- A language to describe digital hardware
- Compatible with EDA tools, as it generates VHDL/Verilog files
- Much more powerful than VHDL, Verilog, and SystemVerilog in its syntax and features
- Much less verbose than VHDL, Verilog, and SystemVerilog
- Not an HLS, nor based on the event-driven paradigm
- Only generates what you asked it in a one-to-one way (no black-magic, no black box)
- Not introducing area/performance overheads in your design (versus a hand-written VHDL/Verilog design)
- Based on the RTL description paradigm, but can go much further
- Allowing you to use Object-Oriented Programming and Functional Programming to elaborate your hardware and verify it
- Free and can be used in the industry without any license



- https://spinalhdl.github.io/SpinalDoc-RTD/
  1st Place

VexRiscv

- https://github.com/SpinalHDL/VexRiscv
  A FPGA friendly 32 bit RISC-V CPU implementation which is written in SpinalHDL.
- Specs

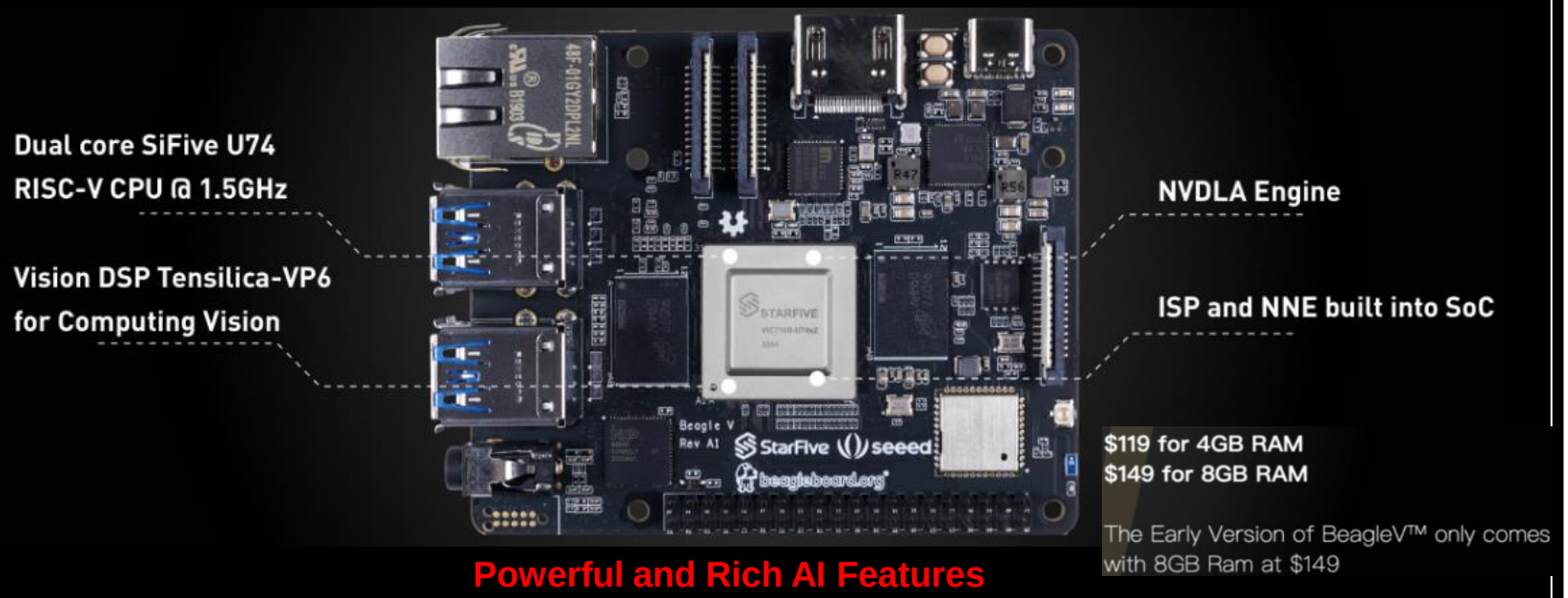
  - RV32I[M][A][F[D]][C] instruction set
  - Pipelined from 2 to 5+ stages ([Fetch*X], Decode, Execute, [Memory], [WriteBack])
  - 1.44 DMIPS/Mhz --no-inline when nearly all features are enabled (1.57 DMIPS/Mhz when the divider lookup table is enabled)
  - Optimized for FPGA, does not use any vendor specific IP block / primitive
  - AXI4, Avalon, wishbone ready
  - Optional MUL/DIV extensions
  - Optional F32/F64 FPU (require data cache for now)
  - Optional instruction and data caches
  - Optional hardware refilled MMU
  - Optional debug extension allowing Eclipse debugging via a GDB >> openOCD >> JTAG connection
  - Optional interrupts and exception handling with Machine, [Supervisor] and [User] modes as defined in the RISC-V Privileged ISA Specification v1.10.
  - Two implementations of shift instructions: single cycle (full barrel shifter) and shiftNumber cycles
  - Each stage can have optional bypass or interlock hazard logic
  - Linux compatible (SoC : https://github.com/enjoy-digital/linux-on-litex-vexriscv)
  - Zephyr compatible
  - FreeRTOS port

- https://riscv.org/announcements/2018/10/risc-v-contest/
  1st Place
- For details, you may refer to my previous talk "Scala-based FOSS EDA on ARM" at the 5th China Functional Programming Meetup 2021 (Shanghai).

4.4 Development boards

4.4.1 BeagleV(abandoned)

- https://beaglev.org/
- The First Affordable RISC-V Computer Designed to Run Linux



Powerful and Rich AI Features

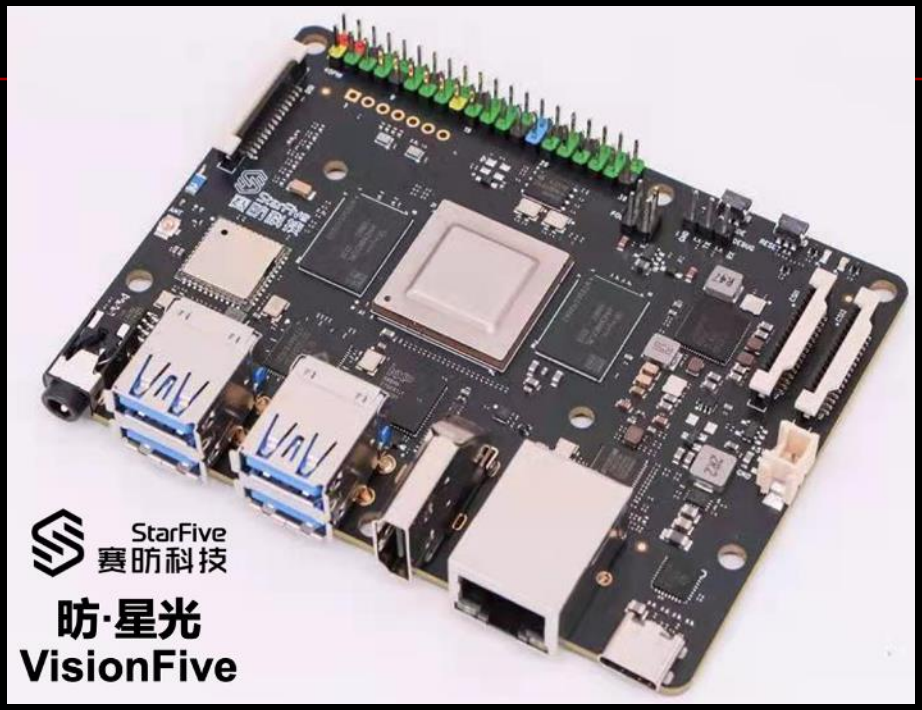
TRULY OPEN SOURCE

Open Source Software | Based on RISC-V Open Architecture | Open Hardware Design

4.4.1.1 Successors

VisionFive

- base on StarFive's JH7100 vision processing SoC



- https://rvspace.org/en/Product/VisionFive/Technical_Documents/VisionFive_Single_Board_Computer_Quick_Start_Guide
- https://rvspace.org/
- https://forum.rvspace.org/t/how-to-purchase-visionfive/37/5
- …

5) IoT Edge

5.1 Overview

- …

5.2 Microcontroller

- https://en.wikipedia.org/wiki/Microcontroller

A **microcontroller** (**MCU** for *microcontroller unit*) is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

In modern terminology, a microcontroller is similar to, but less sophisticated than, a system on a chip (SoC). An SoC may include a microcontroller as one of its components, but usually integrates it with advanced peripherals like a graphics processing unit (GPU), a Wi-Fi module, or one or more coprocessors.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. In the context of the internet of things, microcontrollers are an economical and popular means of data collection, sensing and actuating the physical world as edge devices.

Some microcontrollers may use four-bit words and operate at frequencies as low as 4 kHz for low power consumption (single-digit milliwatts or microwatts). They generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

ARM Cortex-M

- https://en.wikipedia.org/wiki/ARM_Cortex-M

RISC-V Microcontrollers

- …

5.3 RTOS

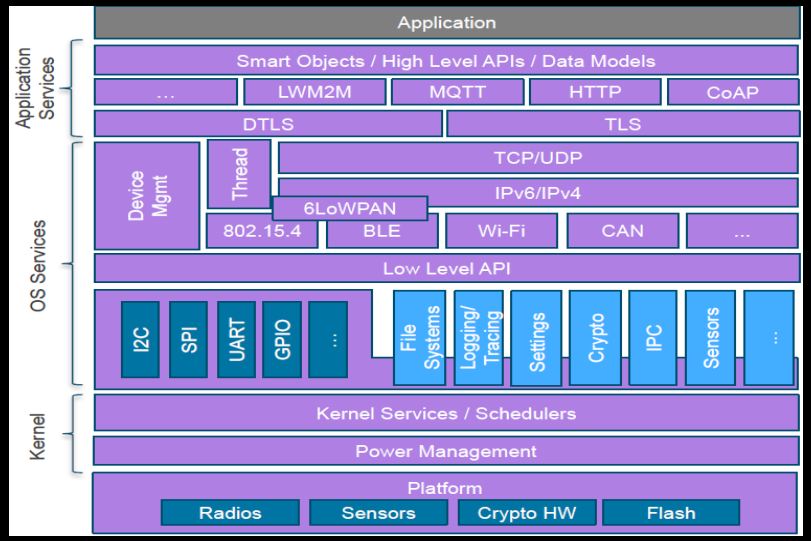
- https://en.wikipedia.org/wiki/Real-time_operating_system

A **real-time operating system** (**RTOS**) is an operating system (OS) for real-time applications that processes data and events that have critically defined time constraints. An RTOS is distinct from a time sharing operating system, such as Unix, which manages the sharing of system resources with a scheduler, data buffers, or fixed task prioritization in a multitasking or multiprogramming environment. Processing time requirements need to be fully understood and bound rather than just kept as a minimum. All processing must occur within the defined constraints. Real-time operating systems are event-driven and preemptive, meaning the OS is capable of monitoring the relevant priority of competing tasks, and make changes to the task priority. Event-driven systems switch between tasks based on their priorities, while time-sharing systems switch the task based on clock interrupts.

Zephyr

- https://www.zephyrproject.org/
- https://github.com/zephyrproject-rtos/zephyr

  The Zephyr kernel supports multiple architectures, including ARM Cortex-M, Intel x86, ARC, Nios II, Tensilica Xtensa, and RISC-V, and a large number of supported boards(350+).
- Architecture



Source: https://www.zephyrproject.org/wp-content/uploads/sites/38/2020/09/zephyr_overview_091520.pdf

5.4 TinyML

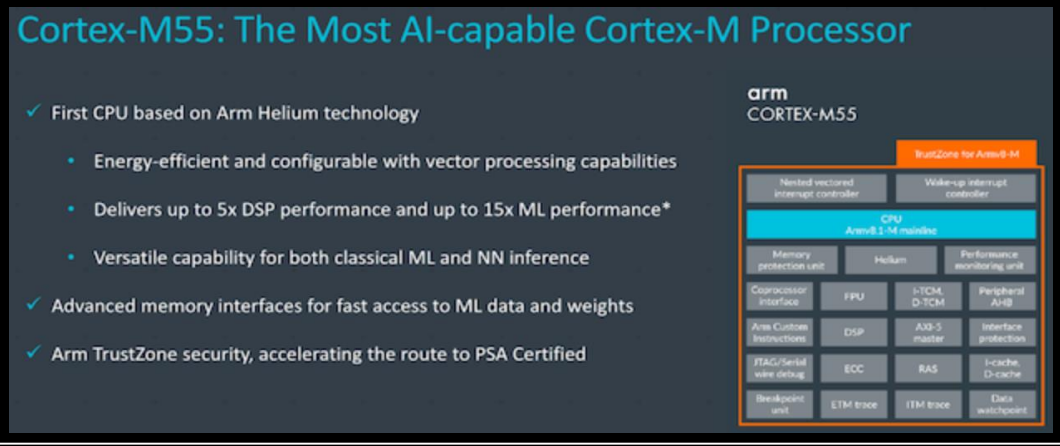
- https://www.tinyml.org/

Tiny machine learning is broadly defined as a fast growing field of machine learning technologies and applications including hardware, algorithms and software capable of performing on-device sensor data analytics at extremely low power, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices.

- https://semiengineering.com/why-tinyml-is-such-a-big-deal/

TinyML on ARM

- https://www.arm.com/blogs/blueprint/tinyml
- Cortex-M55
  https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m55
  First Helium and Custom Instruction capable CPU core



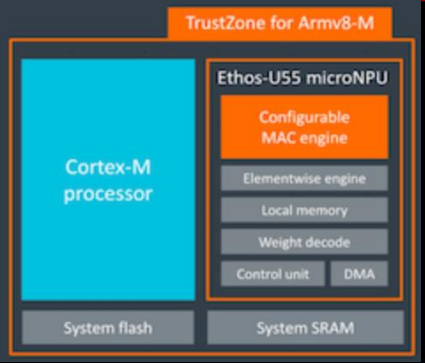
Ethos: MicroNPU for ARM

https://www.arm.com/product-filter?families=ethos%20npus&showall=true

Ethos-U55:

Ethos-U55: The First microNPU for Cortex-M

- ✓ Highest efficiency and small memory footprint
- ✓ 32, 64, 128, or 256 unit multiply-accumulate (MAC) engine
- ✓ Weight decoder and DMA for on-the-fly weight decompression
- ✓ Tooling available for offline optimization
- ✓ Works with a range of Cortex-M processors:
  - Cortex-M55
  - Cortex-M7
  - Cortex-M33
  - Cortex-M4



Key Features	Performance (At 1GHz)	64 to 512 GOP/s
	MACs (8x8)	32, 64, 128, 256
	Utilization on popular networks	Up to 85%
	Data Types	Int-8 and Int-16
	Network Support	CNN and RNN/LSTM
	Winograd Support	No
	Sparsity	Yes
Memory System	Internal SRAM	18 to 50 KB
	External On Chip SRAM	KB to Multi-MB
	Compression	Weights only
	Memory Optimizations	Extended compression, layer/operator fusion
Development Platform	Neural Frameworks	TensorFlow Lite Micro
	Operating Systems	RTOS or bare-metal
	Software Components	TensorFlow Lite Micro Runtime, CMSIS-NN, Optimizer, Driver
	Debug and Profile	Layer-by-layer visibility with PMUs
	Evaluation and Early Prototyping	Performance Model, Cycle Accurate Model, or FPGA Evaluations

6) .Net

6.1 Overview

- https://en.wikipedia.org/wiki/.NET
- https://dotnet.microsoft.com/

- https://docs.microsoft.com/en-us/dotnet/standard/glossary
- https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
- https://en.wikipedia.org/wiki/Windows_Runtime
- https://en.wikipedia.org/wiki/.NET_Framework
- …
- https://github.com/quozd/awesome-dotnet
- https://github.com/thangchung/awesome-dotnet-core

Open Source

- https://dotnet.microsoft.com/platform/open-source
- https://github.com/dotnet
- https://github.com/aspnet
- https://github.com/microsoft
- https://en.wikipedia.org/wiki/Mono_(software)
- ...

CLI

- https://en.wikipedia.org/wiki/Common_Language_Infrastructure

The **Common Language Infrastructure** (**CLI**) is an open specification and technical standard originally developed by Microsoft and standardized by ISO (**ISO/IEC 23271**) and Ecma International (**ECMA 335**)[1][2] that describes executable code and a runtime environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures. This implies it is platform agnostic. The .NET Framework, .NET and Mono are implementations of the CLI. The metadata format is also used to specify the API definitions exposed by the Windows Runtime.[3][4]

Among other things, the CLI specification describes the following four aspects:

The Common Type System (CTS)

A set of data types and operations that are shared by all CTS-compliant programming languages.

The Metadata

Information about program structure is language-agnostic, so that it can be referenced between languages and tools, making it easy to work with code written in a language the developer is not using.

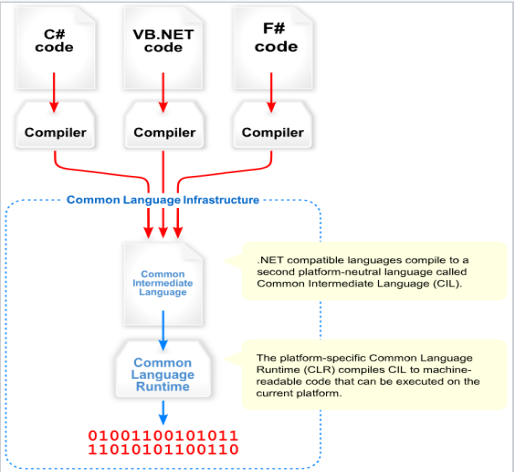
The Common Language Specification (CLS)

The CLI should conform with the set of base rules to which any language targeting, since that language should interoperate with other CLS-compliant languages. The CLS rules define a subset of the Common Type System.

The Virtual Execution System (VES)

The VES loads and executes CLI-compatible programs, using the metadata to combine separately generated pieces of code at runtime.

All compatible languages compile to Common Intermediate Language (CIL), which is an intermediate language that is abstracted from the platform hardware. When the code is executed, the platform-specific VES will compile the CIL to the machine language according to the specific hardware and operating system.

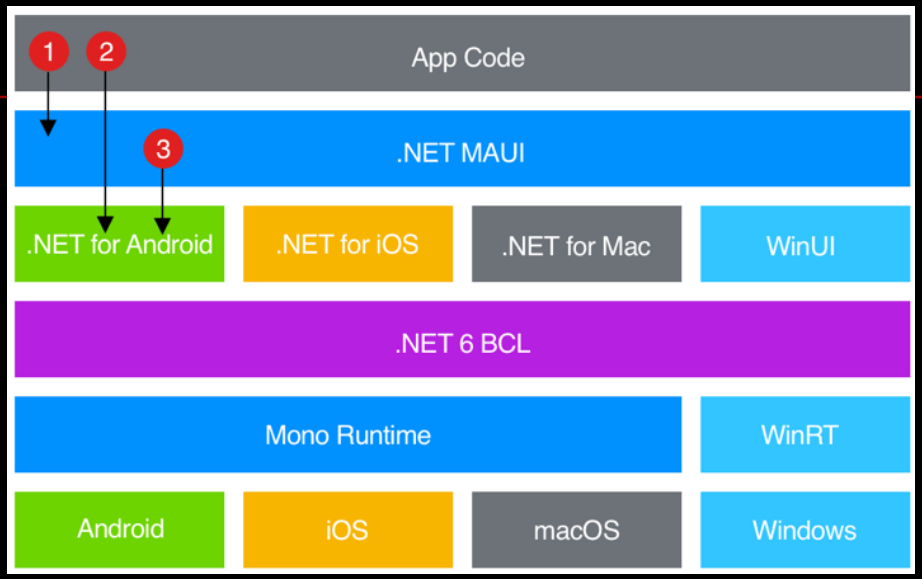


Visual overview of the Common Language Infrastructure (CLI)

- https://en.wikipedia.org/wiki/Common_Intermediate_Language
- https://en.wikipedia.org/wiki/Common_Language_Runtime

.Net 6

- https://rubikscode.net/2021/08/30/net-6-top-6-new-features-in-the-upcoming-net-6-version/



-

Version	Release date	Released with	Latest update	Latest update date	Support ends[19]
.NET Core 1.0	2016-06-27[20]	Visual Studio 2015 Update 3	1.0.16	May 14, 2019	June 27, 2019
.NET Core 1.1	2016-11-16[21]	Visual Studio 2017 Version 15.0	1.1.13	May 14, 2019	June 27, 2019
.NET Core 2.0	2017-08-14[13]	Visual Studio 2017 Version 15.3	2.0.9	July 10, 2018	October 1, 2018
.NET Core 2.1	2018-05-30[14]	Visual Studio 2017 Version 15.7	2.1.30 (LTS)	August 19, 2021	August 21, 2021
.NET Core 2.2	2018-12-04[15]	Visual Studio 2019 Version 16.0	2.2.8	November 19, 2019	December 23, 2019
.NET Core 3.0	2019-09-23[22]	Visual Studio 2019 Version 16.3	3.0.3	February 18, 2020	March 3, 2020
.NET Core 3.1	2019-12-03[23]	Visual Studio 2019 Version 16.4	3.1.21 (LTS)	November 8, 2021	December 3, 2022
.NET 5	2020-11-10[24]	Visual Studio 2019 Version 16.8	5.0.12	November 8, 2021	May 8, 2022
.NET 6	2021-11-08[25]	Visual Studio 2022 Version 17.0	6.0.0 (LTS)	November 8, 2021	November 8, 2024
.NET 7	2022-11 (projected)				May 2024 (projected)
.NET 8	2023-11 (projected)		(will be LTS)		November 2026 (projected)

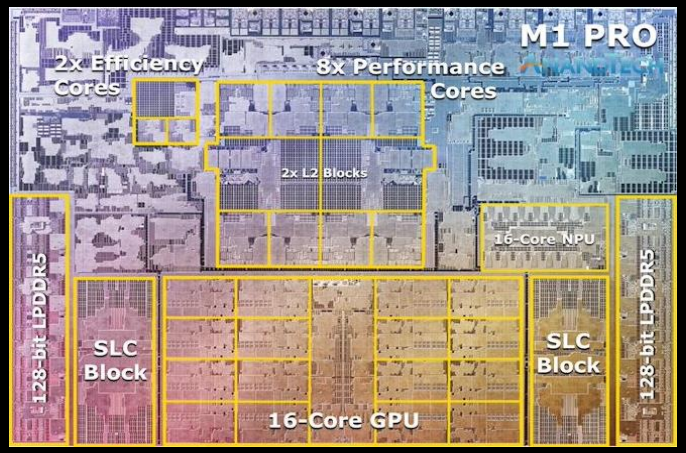
- ...

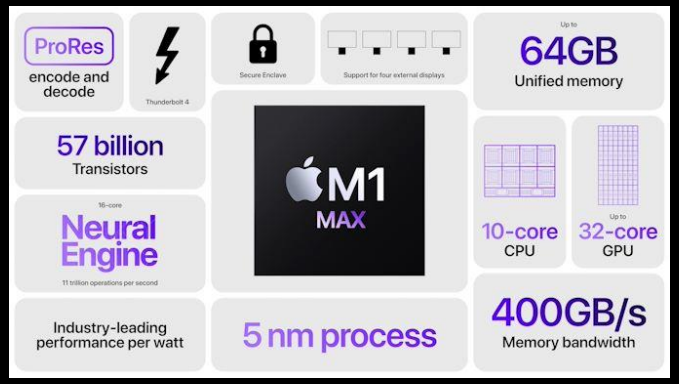
7) Testbed

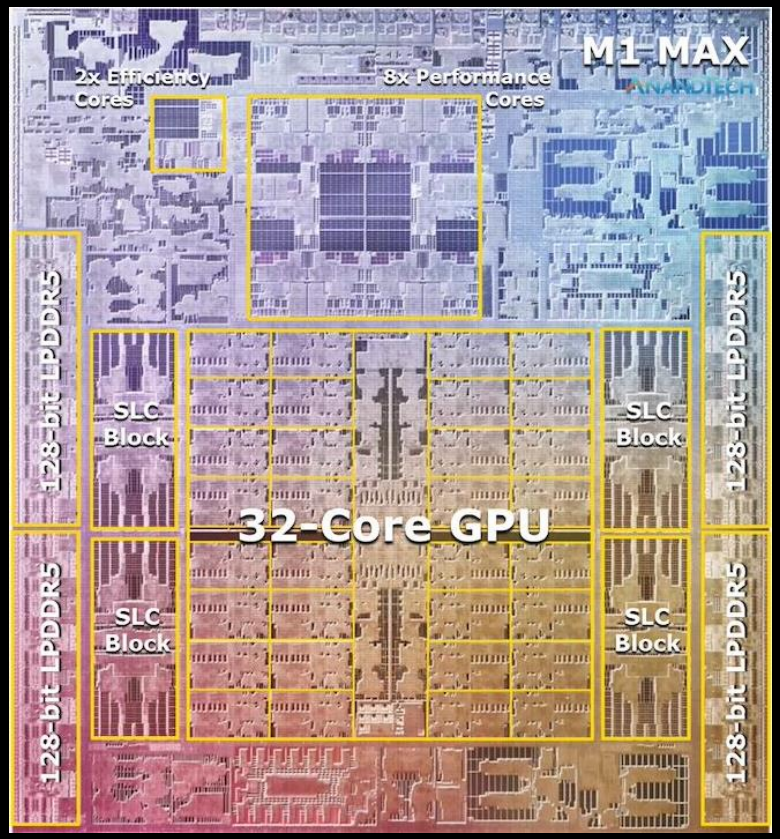
7.1 ARM Ecosystem in 2021

Apple M1 Pro & M1 Max

- https://www.anandtech.com/show/17024/apple-m1-max-performance-review



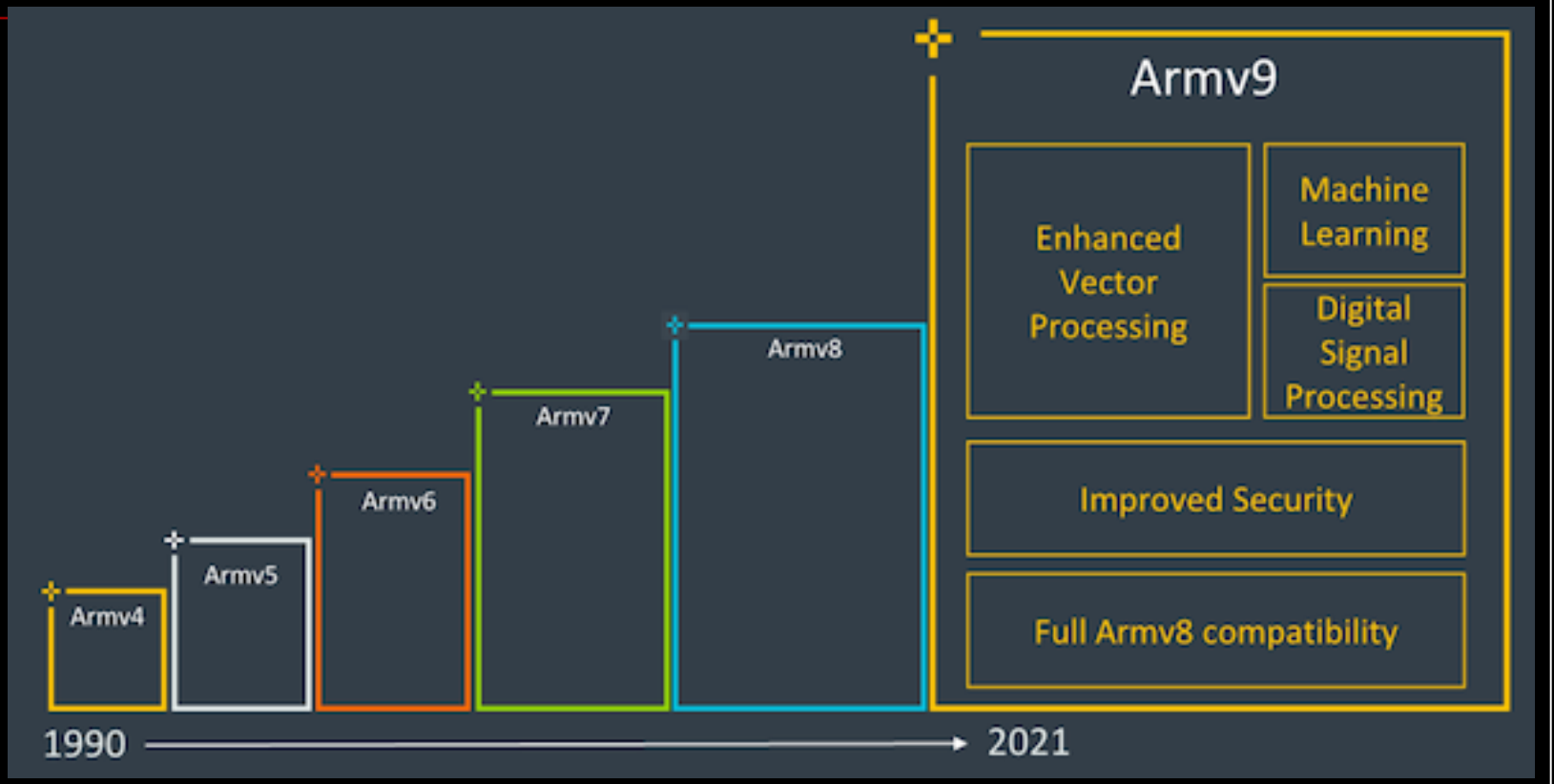


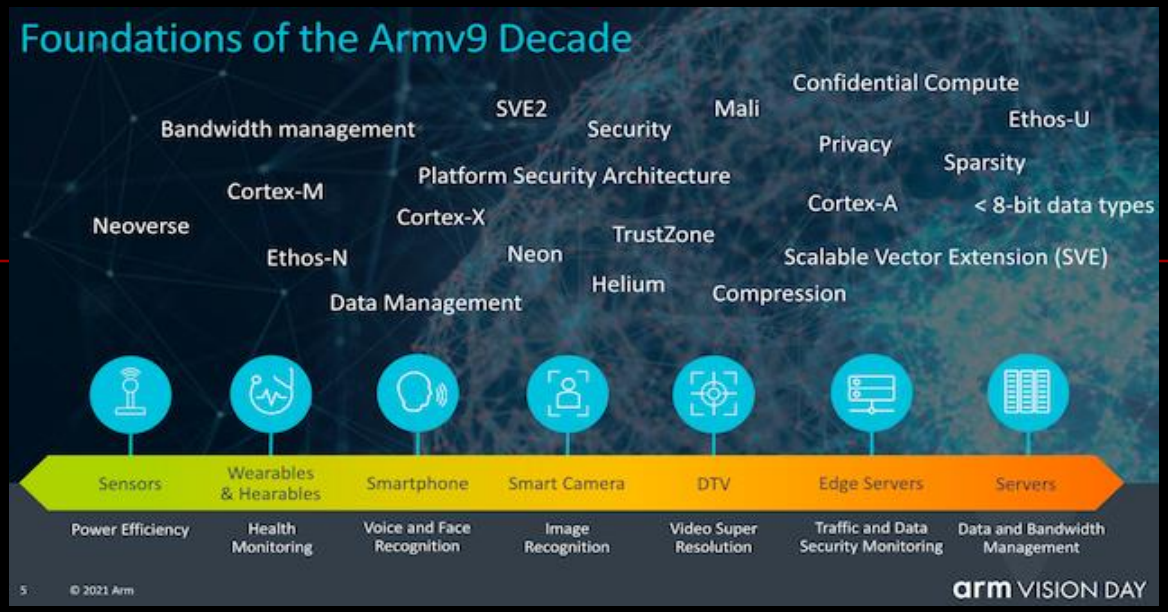


- …

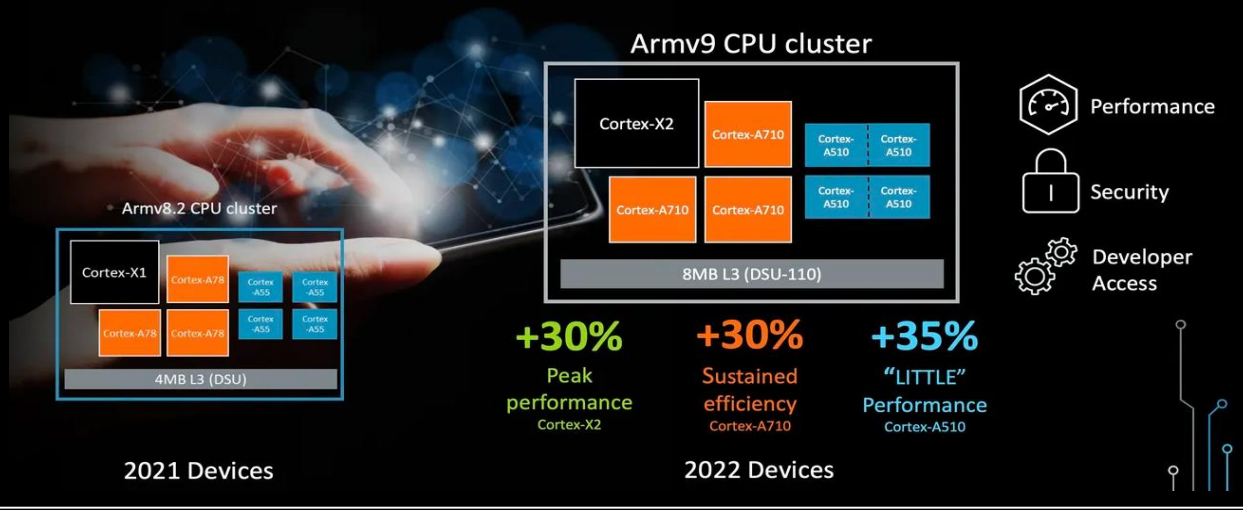
ARM v9

- https://www.arm.com/company/news/2021/03/arms-answer-to-the-future-of-ai-armv9-architecture
- https://www.arm.com/blogs/blueprint/armv9





Armv9 CPU Cluster: A Step Change in Premium Mobile



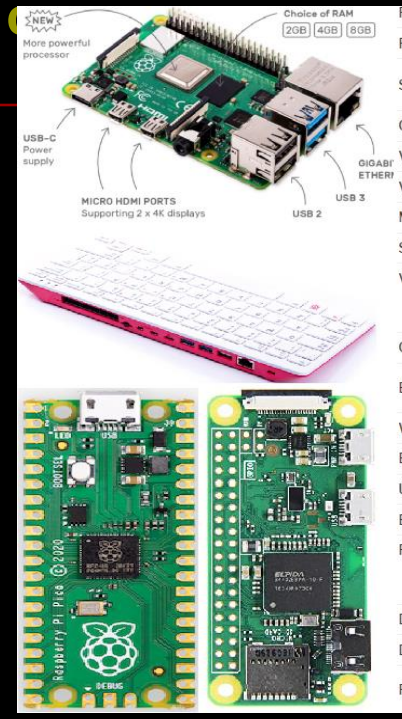
Graviton3

- https://www.nextplatform.com/2021/12/02/aws-goes-wide-and-deep-with-graviton3-server-chip/

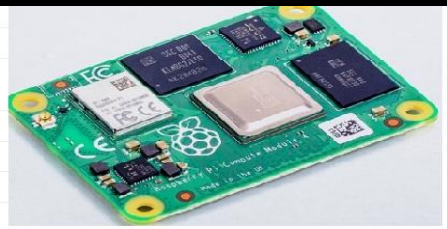
AWS Graviton Server Chips	Graviton	Graviton2	Graviton3
Codename	"Alpine"	"Alpine+"	???
Model	AL73400	ALC12B00	???
Preview	Nov 2018	Nov 2019	Nov 2021
Foundry	TSMC	TSMC	TSMC
Process	16 nm	7 nm	5 nm
Transistors (Billion)	5	30	55
Arm Architecture	ARMv8	ARMv8.2	ARMv8.5
Clock Speed	2.3 GHz	2.5 GHz	2.6 GHz
Core Type	"Maya" Cortex-A72	"Ares" N1	"Perseus" N2
Estimated AWS ECU Performance	44.4	310.8	436.36
BogoMIPS	166.66	243.75	342.22
Cores/Threads	16 / 16	64 / 64	64 / 64
L1 Instruction Cache/Core	48 KB	64 KB	64 KB
L1 Data Cache/Core	32 KB	64 KB	64 KB
Total L1 Cache/Chip	1.25 MB	8 MB	8 MB
L2 Cache/Core	4 x 2 MB *	1 MB	1 MB
Total L2 Cache/Chip	8 MB	64 MB	64 MB
L3 Cache/Core	-	512 KB	1 MB
Total L3 Cache/Chip	-	32 MB	64 MB
Memory Type	DDR4-1600	DDR4-3200	DDR5-4800
Memory Controllers	4	8	8
Memory Bandwith Per Socket	51.2 GB/sec	204.8 GB/sec	300 GB/sec
PCI-Express Generation	3.0	4.0	5.0
PCI-Express Lanes	32	64	32
Sockets	1	1	1
Power (TDP)	95W	110W	100W
ECU / Watt	0.47	2.83	4.36

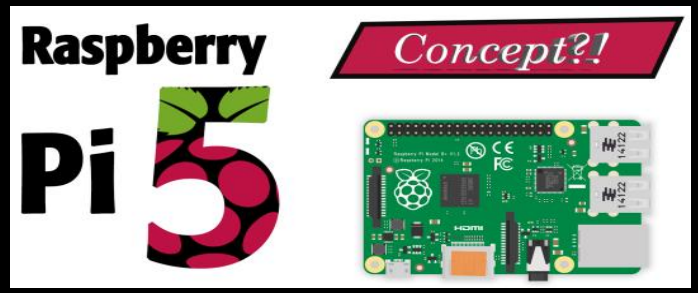
** Two blocks of L2 cache shared by four blocks of CPU tiles on the original Graviton die*

7.1 Raspberry Pi
7.1.1 Overview

- 

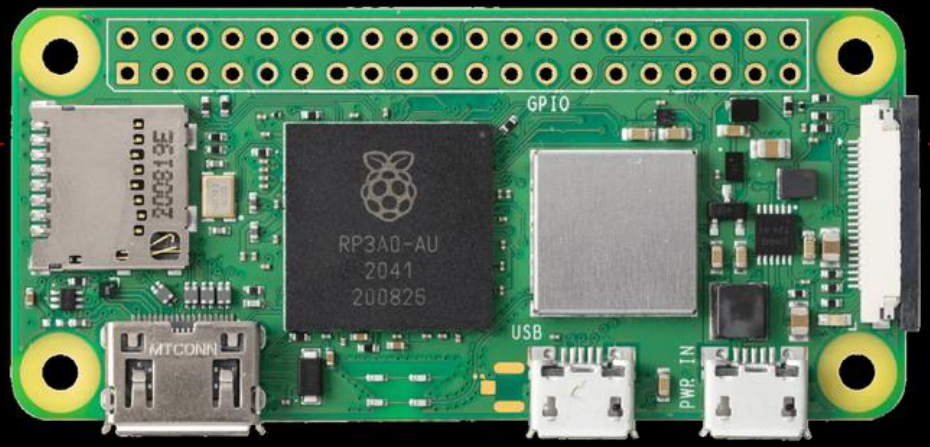
Features/Specs	Raspberry Pi 4B	Raspberry Pi 3 B+
Release date	24th June 2019	14th March 2018
SoC	Broadcom BCM2711 quad-core Cortex-A72 @ 1.5 GHz	Broadcom BCM2837B0 quad-core Cortex-A53 @ 1.4 GHz
GPU	VideoCore VI with OpenGL ES 1.1, 2.0, 3.0	VideoCore IV with OpenGL ES 1.1, 2.0
Video Decode	H.265 4Kp60, H.264 1080p60	H.264 & MPEG-4 1080p30
Video Encode	H.264 1080p30	
Memory	1GB, 2GB, or 4GB LPDDR4	1GB LPDDR2
Storage	microSD card	
Video & Audio Output	2x micro HDMI ports up to 4Kp60 3.5mm AV port (composite + audio) MIPI DSI connector	1x HDMI 1.4 port up to 1080p60 3.5mm AV port (composite + audio) MIPI DSI connector
Camera	MIPI CSI connector	
Ethernet	Native Gigabit Ethernet	Gigabit Ethernet over USB (300 Mbps max.)
WiFi	Dual band 802.11 b/g/n/ac	
Bluetooth	Bluetooth 5.0 + BLE	Bluetooth 4.2 + BLE
USB	2x USB 3.0 + 2x USB 2.0	4x USB 2.0
Expansion	40-pin GPIO header	
Power Supply	5V via USB type-C up to 3A 5V via GPIO header up to 3A Power over Ethernet via PoE HAT	5V via micro USB up to 2.5A 5V via GPIO header up to 3A Power over Ethernet via PoE HAT
Dimensions	85×56 mm	
Default OS	Raspbian (after June 24, 2019)	Raspbian (after March 2018)
Price	$35 (1GB RAM), $45 (2GB RAM), $55 (4GB RAM)	$35 (1GB RAM)

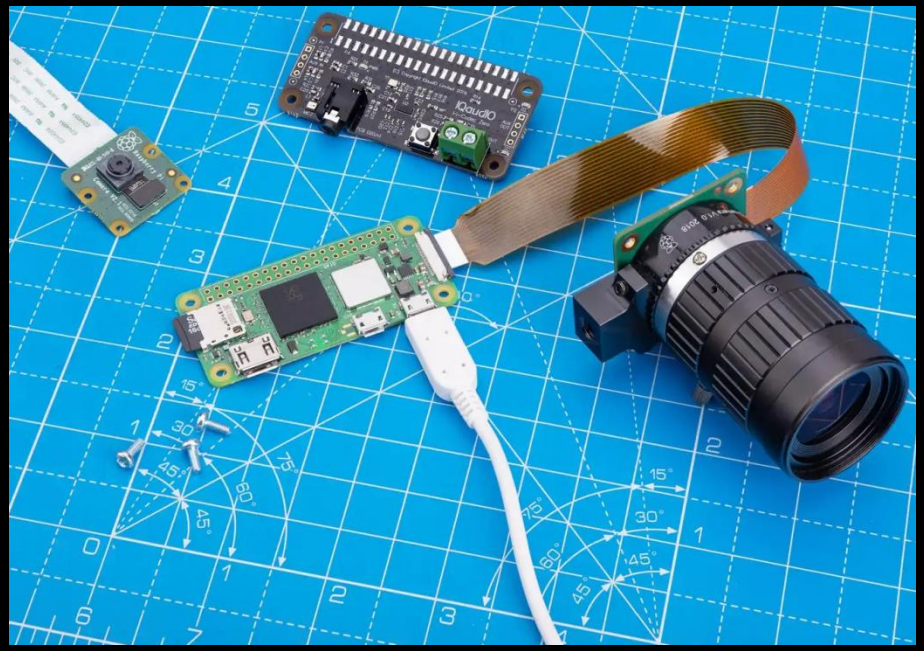


- 

Raspberry Pi Zero 2 W

- https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/





1GHz quad-core 64-bit Arm Cortex-A53 CPU

512MB SDRAM

2.4GHz 802.11 b/g/n wireless LAN

Bluetooth 4.2, Bluetooth Low Energy (BLE), onboard antenna

Mini HDMI port and micro USB On-The-Go (OTG) port

microSD card slot

CSI-2 camera connector

HAT-compatible 40-pin header footprint (unpopulated)

H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30)

OpenGL ES 1.1, 2.0 graphics

Micro USB power

Composite video and reset pins via solder test points

65mm x 30mm

Vulkan

- https://www.raspberrypi.com/news/vulkan-update-version-1-1-conformance-for-raspberry-pi-4/
- https://qengineering.eu/install-vulkan-on-raspberry-pi.html
- https://github.com/karnkaul/rpi4-install-vulkan
- https://archive.fosdem.org/2021/schedule/event/rpi4_vulkan/
- …

OpenCL

- Upcoming…

IPO

- https://www.tomshardware.com/news/raspberry-pi-plans-spring-2022-listing

7.1.2 System

Fedora

- A Linux distribution developed by the community-supported Fedora Project which is sponsored primarily by Red Hat
- https://en.wikipedia.org/wiki/Fedora_(operating_system)
- https://getfedora.org/
- https://alt.fedoraproject.org/alt/
- https://spins.fedoraproject.org/
- https://fedoraproject.org/wiki/Architectures/ARM
- https://fedoramagazine.org/
- https://silverblue.fedoraproject.org/
- Developer friendly!

Fedora 36

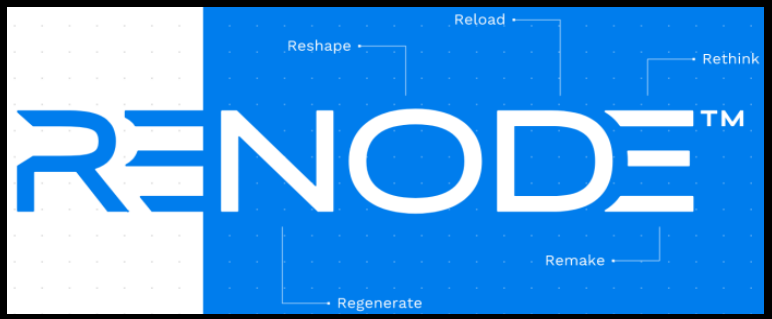
- Enhance the (rpm-)ostree stack to natively support OCI/Docker containers as a transport and delivery mechanism for operating system content. https://fedoraproject.org/wiki/Changes/OstreeNativeContainer
- ...

II. What is Renode

1) Overview

- https://renode.io/

Antmicro's virtual development framework for complex embedded systems.



- https://github.com/renode/renode

Renode was created by Antmicro as a virtual development tool for multinode embedded networks (both wired and wireless) and is intended to enable a scalable workflow for creating effective, tested and secure IoT systems.

With Renode, developing, testing, debugging and simulating unmodified software for IoT devices is fast, cost-effective and reliable.

Supported architectures include:

- ARM Cortex-A and Cortex-M
- x86
- RISC-V
- SPARC
- POWER

- https://renode.io/about/

Strength

Open source hardware-simulation framework for:

- Development of complex software for embedded and IoT systems
- Architectural exploration and research
- Pre-silicon prototyping and HW-SW co-development

Features in brief:

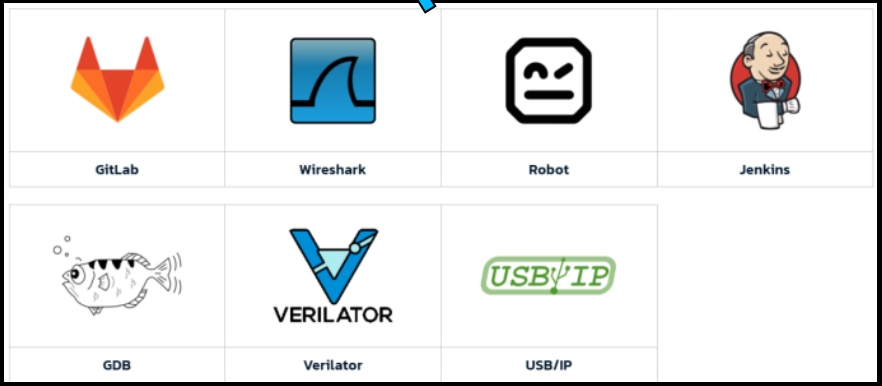
- Plug-and-play building blocks
- Simulates system on many levels - CPUs, SoCs, peripherals, sensors, wired/wireless connection
- Flexible, deterministic and software-agnostic
- Continuous Integration-oriented
- “Batteries included” - lots of demos and binaries

- **full determinism** of execution, shared virtual time
- **transparent & robust debugging**, tracing, analysis, even in multi-node setups
- **easy integration** with your everyday tools, plugins
- **rich model abstractions** with additional functionality "for free" + modular platform description format
- **automated tests and CI integrations**, other collaboration features

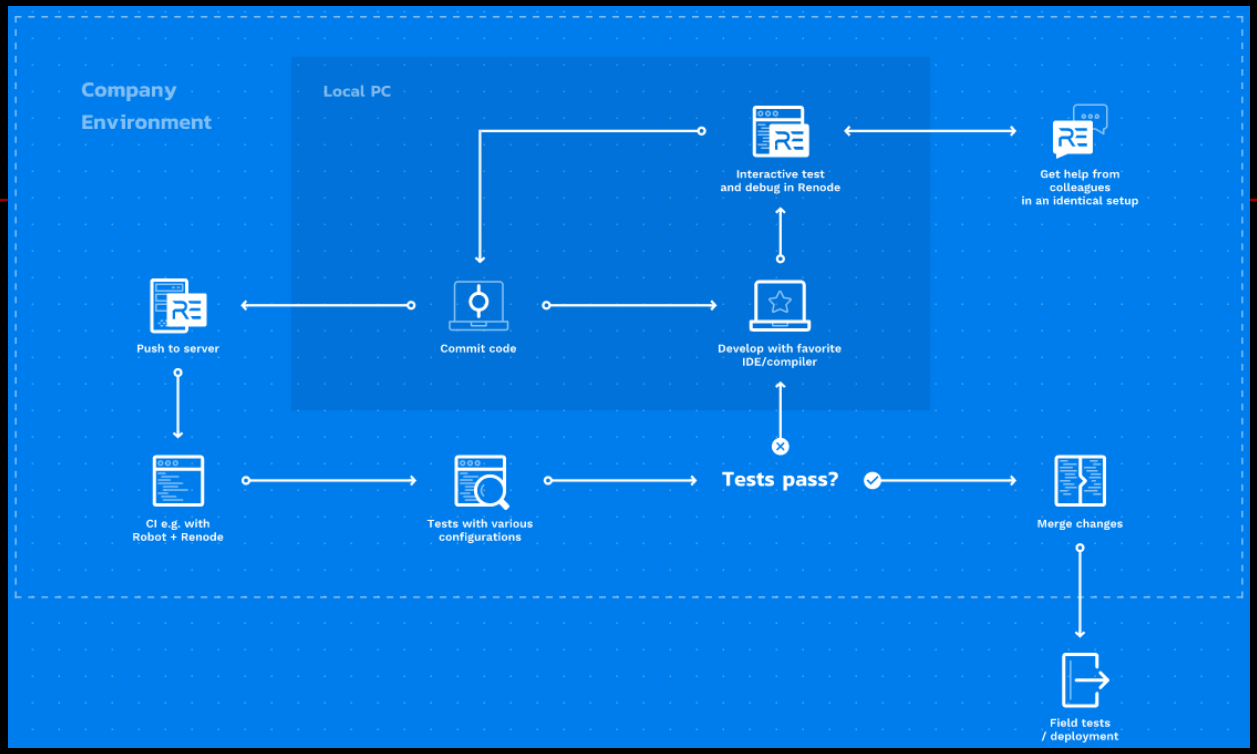
Linux-Capable Platforms & Demos:

- Kendryte (Linux-nommu)
- LiteX/VexRiscv
- HiFive Unleashed (buildroot)
- PolarFire SoC Icicle Kit (Yocto/buildroot)
- Zedboard
- Vybrid
- Versatile
- Versatile Express
- Nvidia Tegra 3

...



Renode Cloud Environment:



Tested SWs:

- Linux
- FreeRTOS
- Zephyr
- RIOT
- TensorFlow Lite
- Mbed
- NuttX

- Tock
- micropython
- Android
- Bare Metal software
- Contiki
- Wolfboot
- eCos
- Redboot
- and more!

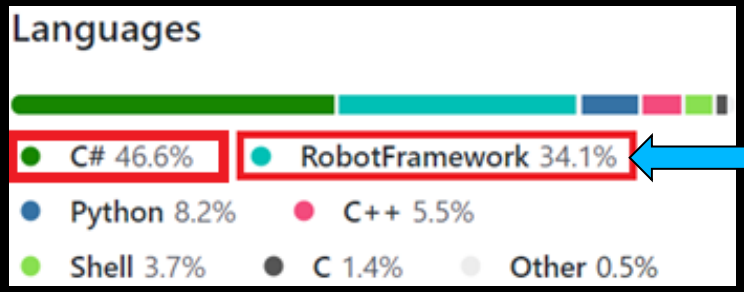
...

2) Architecture & Design

2.1 Overview

2.2.1 Src

- https://github.com/renode/renode



Generic automation framework for acceptance testing and RPA

- depends on Mono(https://www.mono-project.com/) for building and running on Linux
- currently, targets X86 only
- Release

https://dl.antmicro.com/projects/renode/builds/

Stats (includes submodules)

```
[mydev@fedora renode-master]$ git branch
* master
[mydev@fedora renode-master]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 BASH                    5          226          138           38           50
 Batch                   2           63           52            3            8
 C                      65        82364        68963         6463         6938
 C Header               85        54447        49260         2293         2894
 CMake                   3          164          120            5           39
 C#                   3443       525339       385139        82322        57878
 C++                    11         1807         1275          240          292
 Makefile                2           22           14            0            8
 MSBuild               107        12507        12404           64           39
 PowerShell              1           17           13            2            2
 Python                 24         2105         1695           49          361
 ReStructuredText       13         1324          978            0          346
 Shell                  20         1192          909           84          199
 SVG                     1           42           42            0            0
 Plain Text             18      1534475            0      1534352          123
 Visual Studio Sol|     18         2306         2288            0           18
 XAML                    4          278          256            0           22
 XML                   440        65740        65738            2            0
 YAML                    4          237          220            1           16
-------------------------------------------------------------------------------
 Markdown               11         1213            0          862          351
 |- INI                  1            8            7            0            1
 |- Shell                1           20           20            0            0
 (Total)                            1241           27          862          352
===============================================================================
 Total                4277      2285896       589531      1626780        69585
===============================================================================
[mydev@fedora renode-master]$
```

hierarchical structure

```
[mydev@fedora renode-master]$ tree -L 2 -d .
.
├── lib
│   ├── AntShell
│   ├── BigGustave
│   ├── cctask
│   ├── CxxDemangler
│   ├── ELFSharp
│   ├── FdtSharp
│   ├── InpliTftpServer
│   ├── Migrant
│   ├── options-parser
│   ├── Packet.Net
│   ├── termsharp
│   └── xwt
├── platforms
│   ├── boards
│   └── cpus
├── scripts
│   ├── complex
│   ├── multi-node
│   ├── pydev
│   └── single-node
├── src
│   ├── Infrastructure
│   ├── Plugins
│   └── Renode
├── tests
│   ├── metrics-analyzer
│   ├── network-server
│   ├── peripherals
│   ├── platforms
│   └── unit-tests
├── tools
│   ├── building
│   ├── metrics_analyzer
│   └── packaging
└── utils
    └── gdb_compare
```

Model files

- LiteX generates a platform description in csr.csv output file
- Automatic generation from LiteX configuration (CSV)
  - platform (REPL)
  - script (RESC)
  - Zephyr DTS overlay
- Perfect for CI setup

e.g.

```
[mydev@fedora renode-master]$ tree -L 2 platforms/boards
platforms/boards
├── A2_CV32E40P.repl
├── arduino_101-shield.repl
├── arduino_nano_33_ble.repl
├── arty_litex_vexriscv.repl
├── arvsom.repl
├── beaglev_starlight.repl
├── colibri-vf61.repl
├── crosslink-nx-evn.repl
├── eos-s3-qomu.repl
├── eos-s3-quickfeather.repl
├── gr716-devboard.repl
├── ice40up5k-mdp-evn.repl
├── leon3-externals.repl
├── leon3.repl
├── mars_zx3-externals.repl
├── mars_zx3.repl
├── mimxrt1064_evk.repl
├── miv-board-additional-uarts.repl
├── miv-board.repl
├── mpfs-icicle-kit.repl
├── quark_c1000-cc2520.repl
├── silabs
│   ├── brd4162a.repl
│   ├── sltb001a.repl
│   ├── slwstk6220a.repl
│   ├── stk3200.repl
│   ├── stk3600.repl
│   ├── stk3700.repl
│   └── stk3800.repl
├── stm32f072b_discovery.repl
├── stm32f4_discovery-additional_gpios.repl
├── stm32f4_discovery-bb.repl
├── stm32f4_discovery-kit.repl
├── stm32f4_discovery.repl
├── stm32f7_discovery-bb.repl
├── tegra2.repl
├── tegra3.repl
├── tegra_externals.repl
├── versatile.repl
├── vexpress-externals.repl
├── vexpress.repl
├── zedboard-externals.repl
├── zedboard.repl
└── zolertia-firefly.repl
```

```
[mydev@fedora renode-master]$ tree -L 2 scripts/
scripts/
├── complex
│   ├── arduino_nano
│   ├── fomu
│   ├── litex_i2s
│   └── README.rst
├── monitor.py
├── multi-node
│   ├── cc2538
│   ├── quark-c1000-zephyr
│   └── sam_e70.resc
├── pydev
│   ├── counter.py
│   ├── flipflop.py
│   ├── repeater.py
│   ├── rolling-bit.py
│   └── ticker.py
└── single-node
    ├── arty_litex_vexriscv.resc
    ├── arvsom.resc
    ├── beaglev_starlight.resc
    ├── cc2538.resc
    ├── efr32mg.resc
    ├── gr716_zephyr.resc
    ├── hifive_unleashed.resc
    ├── i386.resc
    ├── icicle-kit.resc
    ├── kendryte_k210.resc
    ├── leon3_zephyr.resc
    ├── litex_ibex.resc
    ├── litex_microwatt.resc
    ├── litex_minerva.resc
    ├── litex_netv2_vexriscv_linux.resc
    ├── litex_vexriscv_linux.resc
    ├── litex_vexriscv_micropython.resc
    ├── litex_vexriscv.resc
    ├── litex_vexriscv_sdcard.resc
    ├── litex_vexriscv_smp.resc
    ├── litex_vexriscv_tftp.resc
    ├── litex_vexriscv_tock.resc
    ├── litex_vexriscv_verilated_cfu.resc
    ├── litex_vexriscv_zephyr.resc
    ├── microwatt.resc
    ├── miv.resc
    ├── mpc5567.resc
    ├── murax.resc
    ├── murax_verilated_uart.resc
    ├── nrf52840.resc
    ├── opentitan-earlgrey.resc
    ├── picosoc.resc
    ├── polarfire-soc.resc
    ├── quark_c1000.resc
    ├── quickfeather.resc
    ├── riscv_verilated_liteuart.resc
    ├── riscv_verilated_uartlite.resc
    ├── sam_e70.resc
    ├── sifive_fe310.resc
    ├── stm32f103.resc
    ├── stm32f4_discovery.resc
    ├── stm32f4_tock.resc
    ├── stm32f746_mbed.resc
    ├── stm32f746.resc
    ├── tegra3.resc
    ├── vegaboard_ri5cy.resc
    ├── versatile.resc
    ├── vexpress.resc
    ├── vybrid.resc
    ├── zedboard.resc
    ├── zolertia.resc
    └── zynq_verilated_fastvdma.resc
```

plugins

```
[mydev@fedora renode-master]$ tree -L 5 src/Plugins/
src/Plugins/
├── VerilatorPlugin
│   ├── Connection
│   │   ├── DLLBasedVerilatedPeripheral.cs
│   │   ├── IVerilatedPeripheral.cs
│   │   ├── Protocols
│   │   │   ├── ActionType.cs
│   │   │   └── ProtocolMessage.cs
│   │   └── SocketBasedVerilatedPeripheral.cs
│   ├── Verilated
│   │   └── Peripherals
│   │       ├── BaseDoubleWordVerilatedPeripheral.cs
│   │       ├── CFUVerilatedPeripheral.cs
│   │       └── VerilatedUART.cs
│   ├── VerilatorIntegrationLibrary
│   │   ├── libs
│   │   │   └── socket-cpp
│   │   │       ├── LICENSE
│   │   │       ├── README.md
│   │   │       └── Socket
│   │   ├── src
│   │   │   ├── buses
│   │   │   │   ├── apb3.cpp
│   │   │   │   ├── apb3.h
│   │   │   │   ├── axi.cpp
│   │   │   │   ├── axi.h
│   │   │   │   ├── axilite.cpp
│   │   │   │   ├── axilite.h
│   │   │   │   ├── axi-slave.cpp
│   │   │   │   ├── axi-slave.h
│   │   │   │   ├── bus.h
│   │   │   │   ├── cfu.cpp
│   │   │   │   ├── cfu.h
│   │   │   │   ├── wishbone.cpp
│   │   │   │   └── wishbone.h
│   │   │   ├── peripherals
│   │   │   │   ├── uart.cpp
│   │   │   │   └── uart.h
│   │   │   ├── renode_bus.cpp
│   │   │   ├── renode_bus.h
│   │   │   ├── renode_cfu.cpp
│   │   │   ├── renode_cfu.h
│   │   │   └── renode.h
│   │   └── verilator-integration-library.cmake
│   ├── VerilatorPlugin.cs
│   └── VerilatorPlugin.csproj
└── WiresharkPlugin
    ├── INetworkLogExtensions.cs
    ├── LinkLayer.cs
    ├── Wireshark.cs
    ├── WiresharkPlugin.cs
    ├── WiresharkPlugin.csproj
    └── WiresharkSender.cs
```

Build

lib	[#30034] CLI: Unify plain mode handling	26 days ago
platforms	[#27953] Fixes for NRF52840 GPIO	4 days ago
scripts	[#29660] Add OpenTitan_ROMController	14 days ago
src	[#27953] Fixes for NRF52840 GPIO	4 days ago
tests	[#27953] Fixes for NRF52840 GPIO	4 days ago
tools	[#29305] profiler: Log CPU id in the header	2 months ago
utils/gdb_compare	[FIX] gdb_comapre: Get output line by line	19 days ago
.gitattributes	[DEV] Set git diff format for *.cs files	2 years ago
.gitignore	[DEV] Add .gdb_history to gitignore	last month
.gitmodules	[#23887] Add BigGustave	9 months ago
.renode-root	[#9346] Rework repository structure.	4 years ago
ACKNOWLEDGEMENTS.rst	[FIX] Fix link in ACKNOWLEDGEMENTS	2 years ago
CHANGELOG.rst	[#25391] Add Changelog for v1.12.0	9 months ago
CONTRIBUTING.rst	[DEV] Expand contribution guidelines - issues section	15 months ago
LICENSE	Initial commit of Renode.	5 years ago
README.rst	[FIX] Update name of POWER ISA	2 months ago
Renode.sln	[#25809] building: Do not rebuild TermSharp each time	7 months ago
build.sh	[#25809] building: Do not rebuild TermSharp each time	7 months ago
renode	[#25195] scripts: Do not rely on external realpath	7 months ago
test.sh	[#25505] Prevent running tests with older version of robotframework	6 months ago

- https://renode.readthedocs.io/en/latest/advanced/building_from_sources.html

2.2.2 Submodules

- https://github.com/renode/renode/blob/master/.gitmodules

```
[submodule "src/Infrastructure"]
        path = src/Infrastructure
        url = ../renode-infrastructure.git
[submodule "lib/termsharp"]
        path = lib/termsharp
        url = https://github.com/antmicro/termsharp.git
[submodule "lib/FdtSharp"]
        path = lib/FdtSharp
        url = https://github.com/antmicro/FdtSharp.git
[submodule "lib/Packet.Net"]
        path = lib/Packet.Net
        url = https://github.com/antmicro/Packet.Net.git
[submodule "lib/AntShell"]
        path = lib/AntShell
        url = https://github.com/antmicro/AntShell.git
[submodule "lib/xwt"]
        path = lib/xwt
        url = https://github.com/antmicro/xwt.git
[submodule "lib/ELFSharp"]
        path = lib/ELFSharp
        url = https://github.com/antmicro/elfsharp.git
[submodule "lib/Migrant"]
        path = lib/Migrant
        url = https://github.com/antmicro/Migrant.git
[submodule "lib/cctask"]
        path = lib/cctask
        url = https://github.com/antmicro/cctask.git
[submodule "lib/options-parser"]
        path = lib/options-parser
        url = https://github.com/antmicro/options-parser
[submodule "lib/CxxDemangler"]
        path = lib/CxxDemangler
        url = https://github.com/antmicro/CxxDemangler
[submodule "lib/InpliTftpServer"]
        path = lib/InpliTftpServer
        url = https://github.com/antmicro/InpliTftpServer.git
[submodule "lib/BigGustave"]
        path = lib/BigGustave
        url = https://github.com/antmicro/BigGustave.git
        branch = renode-build
```

```
[mydev@fedora renode-master]$ find . -name "*.git"
./.git
./lib/AntShell/.git
./lib/BigGustave/.git
./lib/CxxDemangler/.git
./lib/ELFSharp/.git
./lib/FdtSharp/.git
./lib/InpliTftpServer/.git
./lib/Migrant/.git
./lib/Packet.Net/.git
./lib/cctask/.git
./lib/options-parser/.git
./lib/termsharp/.git
./lib/termsharp/xwt/.git
./lib/xwt/.git
./src/Infrastructure/.git
./src/Infrastructure/src/Emulator/Cores/tlib/.git
[mydev@fedora renode-master]$
```

renode-infrastructure

- https://github.com/renode/renode-infrastructure

Languages

C# 79.8% C 20.2%

- https://github.com/renode/renode-infrastructure/blob/master/.gitmodules

```
[submodule "src/Emulator/Cores/tlib"]
        path = src/Emulator/Cores/tlib
        url = https://github.com/antmicro/tlib.git
```

- https://github.com/antmicro/tlib

```
[mydev@fedora renode-master]$ tree -L 2 -d src/Infrastructure/src/Emulator/Cores/tlib
src/Infrastructure/src/Emulator/Cores/tlib
├── arch
│   ├── arm
│   ├── i386
│   ├── ppc
│   ├── riscv
│   └── sparc
├── fpu
├── include
└── tcg
    ├── arm
    └── i386
```

2.2.3 Miscs

renode-resources

- https://github.com/renode/renode-resources

```
[mydev@fedora renode-resources-master]$ tree -L 2 .
.
├── fonts
│   ├── RobotoMono-Regular-license
│   └── RobotoMono-Regular.ttf
├── images
│   ├── Renode-logo.png
│   └── windows_setup
├── libraries
│   ├── AsyncIO.dll
│   ├── AsyncIO-license
│   ├── BitMiracle.LibJpeg.NET.dll
│   ├── BitMiracle.LibJpeg.NET-license
│   ├── CommandLine.dll
│   ├── CookComputing.XmlRpcV2.dll
│   ├── CookComputing.XmlRpcV2-license
│   ├── Dynamitey.dll
│   ├── Dynamitey-license
│   ├── IronPython.dll
│   ├── IronPython-license
│   ├── IronPython.Modules.dll
│   ├── IronPython.StdLib.dll
│   ├── libopenlibm-Linux.a
│   ├── Lucene.Net.dll
│   ├── Lucene.Net-license
│   ├── LZ4.dll
│   ├── LZ4-license
│   ├── LZ4n.dll
│   ├── LZ4n-license
│   ├── mcs.dll
│   ├── Microsoft.Dynamic.dll
│   ├── Microsoft.Scripting.dll
│   ├── Microsoft.Scripting.Metadata.dll
│   ├── Mono.Cecil.dll
│   ├── Mono.Cecil-license
│   ├── Mono.Linq.Expressions.dll
│   ├── Mono.Linq.Expressions-license
│   ├── Mono.TextTemplating.dll
│   ├── Mono.TextTemplating-license
│   ├── Moq.dll
│   ├── Moq-license
│   ├── NetMQ.dll
│   ├── NetMQ-license
│   ├── Nini.dll
│   ├── Nini-license
│   ├── openlibm-license
│   ├── Sprache.dll
│   ├── Sprache-license
│   └── Sprache.xml
├── llvm
│   ├── 32_libLLVM-3.4.so
│   ├── 32_libLLVM.dll
│   ├── 32_libLLVM.dylib
│   ├── 64_libLLVM-3.4.so
│   ├── 64_libLLVM.dll
│   ├── 64_libLLVM.dylib
│   ├── libllvm-disas.a
│   ├── libllvm-disas.dll
│   ├── libllvm-disas.dylib
│   └── libllvm-disas.so
├── README
├── styles
│   └── robot.css
└── tools
    ├── nunit3
    ├── nunit-agent.exe
    ├── nunit-console.exe
    ├── nunit-console.exe.config
    ├── nunit-console-runner.dll
    ├── nunit.core.dll
    ├── nunit.core.interfaces.dll
    ├── nunit.framework.dll
    ├── nunit-license
    └── nunit.util.dll
```

```
[mydev@fedora renode-resources-master]$ file llvm/64_libLLVM.dll
llvm/64_libLLVM.dll: PE32+ executable (DLL) (GUI) x86-64, for MS Windows
[mydev@fedora renode-resources-master]$
[mydev@fedora renode-resources-master]$ file llvm/libllvm-disas.so
llvm/libllvm-disas.so: ELF 64-bit LSB shared object, x86-64, version 1 (GNU/Linux), dynamically linked, BuildID[sha1]=c8f60567626ff3895210
827f6df6833f45ef8949, not stripped
[mydev@fedora renode-resources-master]$ file libraries/AsyncIO.dll
libraries/AsyncIO.dll: PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
[mydev@fedora renode-resources-master]$
```

renode-docker

- https://github.com/renode/renode-docker

Dockerfile	Update to Renode 1.12.0	7 months ago
Dockerfile.min	Minor whitespace fixes	4 months ago
README.rst	Small README changes	4 months ago

Renode Docker image

This repository contains a Dockerfile for Renode, available on Docker Hub as `antmicro/renode`.

The image is based on Ubuntu 20.04.

Renode is installed from a release Debian package.

To build the image run `docker build --build-arg userId=`id -u` --build-arg groupId=`id -g` -t renode .`.

The image will start Renode in interactive mode by default.

Renode minimized container image

This Dockerfile uses a multistage build with runtime image based on mono:slim to minimize runtime container components and static image size. By default if no version is passed via --build-arg RENODE_VERSION=x.y.z it will build 1.12.0.

Note on `Podman` replacing `Docker`: https://podman.io/whatis.html

Static image size comparison:

Ubuntu base ~805 MB

Mono:slim ~311 MB

Build the image

```
podman build -t renode_min_test ./ -f Dockerfile.min
```

Run the image locally

```
podman run -p 1234:1234 localhost/renode_min_test
```

Run the image as service

```
podman run -d -p 1234:1234 localhost/renode_min_test
```

Connect to the running interactive instance of Renode

```
telnet localhost 1234
```

Build a specific version of Renode

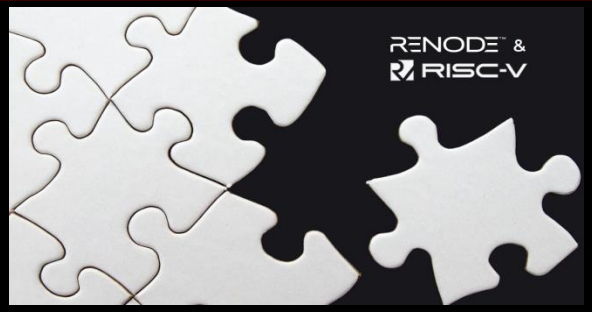
```
podman build -t renode_min_test ./ -f Dockerfile.min --build-arg RENODE_VERSION=1.11.0
```

For more information, visit the Renode website, Renode documentation and Docker documentation.

III. Renode for RISC-V development

1) Built-in Support

1.1 Official

- 

RENODE™ RISC-V®

- https://riscv.org/announcements/2021/12/risc-v-celebrates-incredible-year-of-growth-and-progress-ratifying-multiple-technical-specifications-launching-new-education-programs-and-accelerating-broad-industry-adoption/

Notable achievements of RISC-V adoption in 2021 include:

- Antmicro added support for the RVV 1.0 vector instructions to its open source Renode simulation framework, allowing users to enhance their machine learning development experience in a purely virtual environment. RISC-V Vector ISA support will be one of the highlights of the upcoming Renode 1.13 release.

- https://riscv.org/blog/2021/06/antmicro-open-source-portal-launched/
- https://riscv.org/blog/2021/03/bringing-the-benefits-of-risc-v-and-renode-to-the-very-efficient-deep-learning-in-iot-project/
- …

1.2 LiteX

- https://github.com/enjoy-digital/litex/wiki/Generate-Renode-simulation
  LiteX provides a script for automatic generation of Renode scripts from the json file containing configuration of the LiteX SoC:
  ~~https://github.com/enjoy-digital/litex/blob/master/litex/tools/litex_json2renode.py~~
- Usage

First, build your LiteX platform with `--csr-json csr.json` switch, e.g.:

```
python3 litex/boards/targets/arty.py --cpu-type vexriscv --with-ethernet --csr-json csr.json
```

Now, use the generated configuration file as an input for `litex_json2renode.py`:

```
./litex_json2renode.py csr.json \
    --resc litex.resc \
    --repl litex.repl
    --bios-binary soc_ethernetsoc_arty/software/bios/bios.bin
```

This will generate two files:

- `litex.repl` - platform definition file, containing information about all the peripherals and their configuration,
- `litex.resc` - Renode script file, allowing to easily run the simulation of the generated platform.

Finally, you can run the simulation by executing the command:

```
renode litex.resc
```

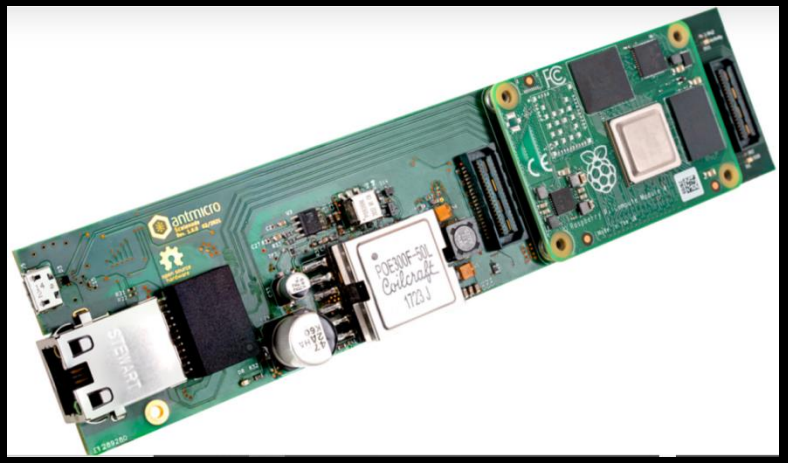
- ...
- Abandoned: https://github.com/litex-hub/litex-renode

1.3 Antmicro

- https://opensource.antmicro.com/
- https://github.com/antmicro
- https://antmicro.com/technologies/risc-v/
- …

HW

- https://antmicro.com/blog/2021/04/scalenode-server-oriented-raspberry-pi4-baseboard/



A scalable compute swarm

Built with server applications in mind, Scalenode is effectively a baseboard for Raspberry Pi 4 Compute Module offering Gigabit Etherner with PoE and several expansion connectors in a server-oriented form factor. It features an M.2 key M PCIe connector for SSD storage or add-on boards that can be used for AI acceleration, e.g. with Google Coral accelerator. To ensure passive cooling of the compute node we designed a custom enclosure, which will also be released as open source. Thanks to its small form factor, as many as 18 Scalenode baseboards fit in a 1U server rack, providing a substantial amount of scalable compute.

Scalenode joins a wide range of open hardware by Antmicro, including server-oriented solutions like the Artix-7 FPGA-based and Lattice ECP5 FPGA-based DC-SCM designs recently adopted by the LibreBMC group in OpenPOWER.

- https://github.com/antmicro/scalenode
- …

2) Getting Started

Official Guide

- https://renode.readthedocs.io/en/latest/introduction/installing.html
- https://renode.readthedocs.io/en/latest/introduction/using.html
- https://renode.readthedocs.io/en/latest/introduction/demo.html
- …

Miscs

- https://blog.y2kbugger.com/baremetal-riscv-renode-1.html
- https://blog.y2kbugger.com/baremetal-riscv-renode-2.html
- …

3) Demos

Official Guide

- https://risc-v-getting-started-guide.readthedocs.io/en/latest/

Introduction

- Introduction
  - About RISC-V
  - Contributing

Zephyr

- Zephyr
  - Why Zephyr?
- Getting Zephyr
- Running Zephyr on QEMU
  - Setting up the environment
  - Compiling an example
  - Running an example
- Running Zephyr on SiFive HiFive1
  - Setting up the enviroment
  - Getting tools for HiFive1 board
  - Compiling an example
  - Flashing
- Running Zephyr on LiteX/VexRiscv on Avalanche board with Microsemi PolarFire FPGA
  - Building your system
  - Running

Linux

- Linux
- Running 64- and 32-bit RISC-V Linux on QEMU
  - Prerequisites
  - Getting the sources
  - Running
- Running 64-bit RISC-V Linux on SiFive HiFive Unleashed
  - Prerequisites
  - Getting the sources
  - Building
  - Flashing
  - Running
- Running 32-bit Linux on LiteX/VexRiscv on Avalanche board with Microsemi PolarFire FPGA
  - Prerequisites
  - Getting the sources
  - Building
  - Running

3.1 Running Linux on LiteX/VexRiscv

- on Win10 for X64

Renode

```
13:31:54.9859 [INFO] Loaded monitor commands from: D:\DevSW\SIM-Modeling\Renode\Portable\v1.11\./scripts/monitor.py
13:32:15.2520 [INFO] System bus created.
13:34:00.9454 [INFO] litex-vexriscv-linux: Machine started.
```

Renode

```
RENODE™

Renode, version 1.11.0.28399 (ad7ab75a-202010221522)

(monitor) mach create "litex-vexriscv-linux"
(litex-vexriscv-linux) machine LoadPlatformDescription @platforms/cpus/litex_vexriscv_linux.repl
(litex-vexriscv-linux) sysbus LoadBinary @image-litex-vexriscv-avalanche-linux 0xc0000000
(litex-vexriscv-linux) sysbus LoadBinary @rootfs-litex-vexriscv-avalanche-linux.cpio 0xc0800000
(litex-vexriscv-linux) sysbus LoadFdt @devicetree-litex-vexriscv-avalanche-linux.dtb 0xc1000000
(litex-vexriscv-linux) sysbus LoadBinary @emulator-litex-vexriscv-avalanche-linux.bin 0x20000000
(litex-vexriscv-linux) sysbus.cpu PC 0x20000000
(litex-vexriscv-linux) showAnalyzer sysbus.uart
(litex-vexriscv-linux) start
Starting emulation...
(litex-vexriscv-linux)
```

litex-vexriscv-linux:sysbus.uart

```
Starting network: ip: socket: Function not implemented
ip: socket: Function not implemented
FAIL

Welcome to Buildroot
buildroot login: root

   __   _                                  __   _ __      _  __   _   __          ___  _
  / /  (_)__  __ ____ _________  ___  ____/ /  (_) /____ | |/_/__| | / /____ __ / _ \(_)__ _____  _
 / /__/ / _ \/ // /\ \ /___/ _ \/ _ \/___/ /__/ / __/ -_)>  </___/ |/ / -_) \ // , _/ (_-</ __/ |/
/____/_/_//_/\_,_//_\_\    \___/_//_/   /____/_/\__/\__/_/|_|    |___/\__/_\_\/_/|_/_/___/\__/|___/

                    32-bits VexRiscv CPU with MMU integrated in a LiteX SoC


login[48]: root login on 'hvc0'
root@buildroot:~#
```

3.2 Running Zephyr on LiteX/VexRiscv

- on Win10 for X64

Renode

```
21:41:33.1944 [INFO] Loaded monitor commands from: D:\DevSW\SIM-Modeling\Renode\Portable\v1.11\./scripts/monitor.py
21:42:12.9745 [INFO] System bus created.
21:43:44.0484 [INFO] sysbus: Loading segment of 69812 bytes length at 0x40000000.
21:43:44.0963 [INFO] cpu: Setting PC value to 0x40000000.
21:44:17.6563 [INFO] litex-vexriscv: Machine started.
```

Renode

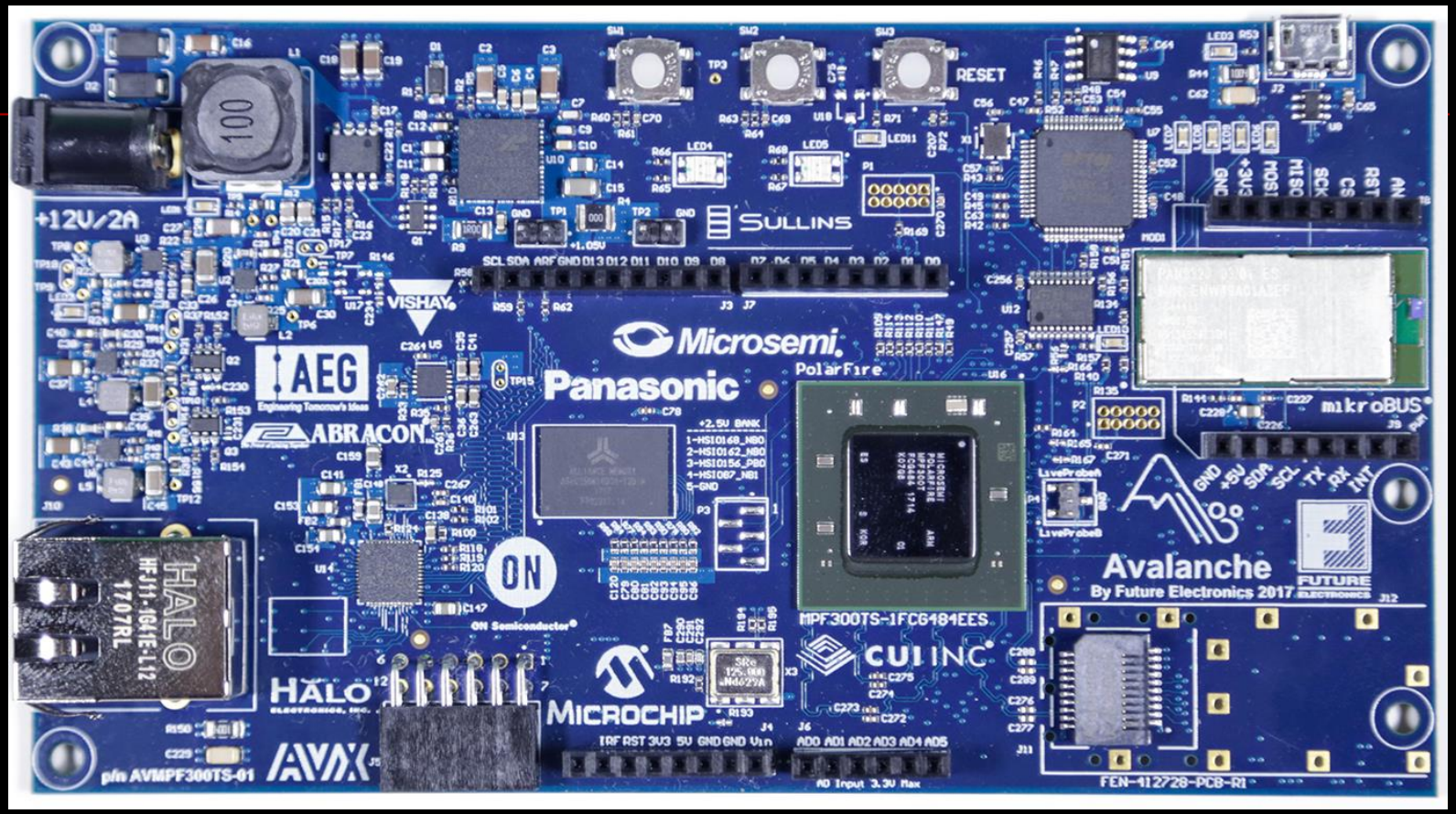
```
RENODE™

Renode, version 1.11.0.28399 (ad7ab75a-202010221522)

(monitor) mach create "litex-vexriscv"
(litex-vexriscv) machine LoadPlatformDescription @platforms/cpus/litex_vexriscv.repl
(litex-vexriscv) sysbus LoadELF @litex_vexriscv_zephyr_shell.elf
(litex-vexriscv) showAnalyzer sysbus.uart
(litex-vexriscv) start
Starting emulation...
(litex-vexriscv)
```

litex-vexriscv:sysbus.uart

```
uart:~$ help
Please press the <Tab> button to see all available commands.
You can also use the <Tab> button to prompt or auto-complete all commands or its subc
ommands.
You can try to call commands with <-h> or <--help> parameter for more information.
Shell supports following meta-keys:
Ctrl+a, Ctrl+b, Ctrl+c, Ctrl+d, Ctrl+e, Ctrl+f, Ctrl+k, Ctrl+l, Ctrl+u, Ctrl+w
Alt+b, Alt+f.
Please refer to shell documentation for more details.
uart:~$
uart:~$ kernel version
Zephyr version 1.14.0
uart:~$
```

3.3 Avalanche board

- Future Electronics Avalanche Board



- Microsemi PolarFire SoC
  Multi-Core RISC-V SoC FPGA…

Running 32-bit Linux on LiteX/VexRiscv on Avalanche board

Renode

```
13:31:54.9859 [INFO] Loaded monitor commands from: D:\DevSW\SIM-Modeling\Renode\Portable\v1.11\./scripts/monitor.py
13:32:15.2520 [INFO] System bus created.
13:34:00.9454 [INFO] litex-vexriscv-linux: Machine started.
```

Renode

RENODE™

```
Renode, version 1.11.0.28399 (ad7ab75a-202010221522)

(monitor) mach create "litex-vexriscv-linux"
(litex-vexriscv-linux) machine LoadPlatformDescription @platforms/cpus/litex_vexriscv_linux.repl
(litex-vexriscv-linux) sysbus LoadBinary @image-litex-vexriscv-avalanche-linux 0xc0000000
(litex-vexriscv-linux) sysbus LoadBinary @rootfs-litex-vexriscv-avalanche-linux.cpio 0xc0800000
(litex-vexriscv-linux) sysbus LoadFdt @devicetree-litex-vexriscv-avalanche-linux.dtb 0xc1000000
(litex-vexriscv-linux) sysbus LoadBinary @emulator-litex-vexriscv-avalanche-linux.bin 0x20000000
(litex-vexriscv-linux) sysbus.cpu PC 0x20000000
(litex-vexriscv-linux) showAnalyzer sysbus.uart
(litex-vexriscv-linux) start
Starting emulation...
(litex-vexriscv-linux)
```

litex-vexriscv-linux:sysbus.uart

```
Starting network: ip: socket: Function not implemented
ip: socket: Function not implemented
FAIL

Welcome to Buildroot
buildroot login: root

   __   _                                  __   _ __      _ __   _   __          ___  _
  / /  (_)__  __ ____ _________  ___  ____/ /  (_) /____ | |/_/__| | / /____ __ / _ \(_)__ _____  _
_
 / /__/ / _ \/ // /\ \ /___/ _ \/ _ \/___/ /__/ / __/ -_)>  </___/ |/ / -_) \ // , _/ (_-</ __/ |/
/
/____/_/_//_/\_,_//_\_\    \___/_//_/   /____/_/\__/\__/_/|_|    |___/\__/_\_\/_|_/_/___/\__/|___/

                    32-bits VexRiscv CPU with MMU integrated in a LiteX SoC


login[48]: root login on 'hvc0'
root@buildroot:~#
```

3.4 BeagleV

- https://antmicro.com/blog/2021/05/linux-on-beaglev-starlight-in-renode/

```
00:24:31.5069 [INFO] Download done.
00:24:32.7688 [INFO] System bus created.
00:24:34.4132 [INFO] sysbus: Loading segment of 74752 bytes length at 0x80000000.
00:24:34.4281 [INFO] sysbus: Loading segment of 510472 bytes length at 0x80020000.
00:24:34.4630 [INFO] U74_2: Setting PC value to 0x80000000.
00:24:34.4640 [INFO] U74_1: Setting PC value to 0x80000000.
00:24:35.0204 [INFO] BeagleV: Machine started.
00:24:35.1672 [WARNING] sysbus: [U74_2: 0x800025F2] (tag: 'SDIO') ReadDoubleWord from non existing peripheral at 0x1000002C, returning 0x0.
```

Renode

```
RENODE™
Renode, version 1.12.0.33058 (c15700ab-202105181821)

(monitor) start @scripts/single-node/beaglev_starlight.resc
Script loaded. Now start with with the 'start' command.

(BeagleV)
```

BeagleV:sysbus.uart3

```
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting mdev... OK
modprobe: can't change directory to '/lib/modules': No such file or directory
Saving random seed: [   17.303863] random: dd: uninitialized urandom read (512 bytes read)
OK
Starting system message bus: [   17.375023] random: dbus-uuidgen: uninitialized urandom read
 (12 bytes read)
[   17.375395] random: dbus-uuidgen: uninitialized urandom read (8 bytes read)
done
Starting network: [   17.634353] stmmaceth 10020000.gmac eth0: validation of rgmii-txid with
 support 0000000,00000000,00006280 and advertisement 0000000,00000000,00006280 failed: -22
[   17.634833] stmmaceth 10020000.gmac eth0: stmmac_open: Cannot attach to PHY (error: -22)
ip: SIOCSIFFLAGS: Invalid argument
FAIL
Starting dropbear sshd: OK
Starting DHCP server: FAIL

Welcome to Buildroot
buildroot login:
```

```
00:24:35.3922 [WARNING] sysbus: [U74_2: 0x80021A32] (tag: 'CLKGEN_CSR') WriteDoubleWord to non existing peripheral at 0x118000B8, value 0x80000000.
00:24:35.3922 [WARNING] sysbus: [U74_2: 0x80021A36] (tag: 'CLKGEN_CSR') ReadDoubleWord from non existing peripheral at 0x118000BC, returning 0x0.
00:24:35.3932 [WARNING] sysbus: [U74_2: 0x80021A44] (tag: 'CLKGEN_CSR') WriteDoubleWord to non existing peripheral at 0x118000BC, value 0x80000000.
00:24:35.3942 [WARNING] sysbus: [U74_2: 0x80021AD8] (tag: 'SYSCTRL') ReadDoubleWord from non existing peripheral at 0x11850050, returning 0x0.
00:24:35.3972 [WARNING] sysbus: [U74_2: 0x80021AE8] (tag: 'SYSCTRL') WriteDoubleWord to non existing peripheral at 0x11850050, value 0x1.
00:24:35.3972 [WARNING] sysbus: [U74_2: 0x80021AEA] (tag: 'CLKGEN_CSR') ReadDoubleWord from non existing peripheral at 0x118000C0, returning 0x0.
00:24:35.3972 [WARNING] sysbus: [U74_2: 0x80021B02] (tag: 'CLKGEN_CSR') WriteDoubleWord to non existing peripheral at 0x118000C0, value 0x80000000.
00:24:35.3982 [WARNING] sysbus: [U74_2: 0x80021B06] (tag: 'CLKGEN_CSR') ReadDoubleWord from non existing peripheral at 0x118000DC, returning 0x0.
```

3.5 Note

- On page 25, you should replace the command "cpu PC 0x20000000" with "sysbus.cpu PC 0x20000000" if you are using the latest versions of Renode.
- For Linux on LiteX/VexRISCV, the needed files like device tree and emulator can be downloaded from the official site, while there are no downloadable binaries for kernel and rootfs, so we have to build them from source by ourselve – the most important step is to apply a patch(on page 24, and actually, that patch is at https://github.com/riscv/risc-v-getting-started-guide/blob/master/source/files/0001-Add-Litex-VexRiscv-support.patch) to the buildroot codebase(ensure that your network is in good condition, unless the build procedure may take too much time as a few software packages should be downloaded from the internet).
- …

4) Co-simulation

Official Guide

- https://renode.readthedocs.io/en/latest/tutorials/verilator-cosimulation.html
- https://antmicro.com/blog/2021/06/advanced-co-simulation-with-renode-and-verilator/
- …

5) Debugging

Official Guide

- https://renode.readthedocs.io/en/latest/basic/logger.html
- https://renode.readthedocs.io/en/latest/basic/metrics.html
- https://renode.readthedocs.io/en/latest/debugging/gdb.html
- https://antmicro.com/blog/2021/04/renode-1-12-release/
- …

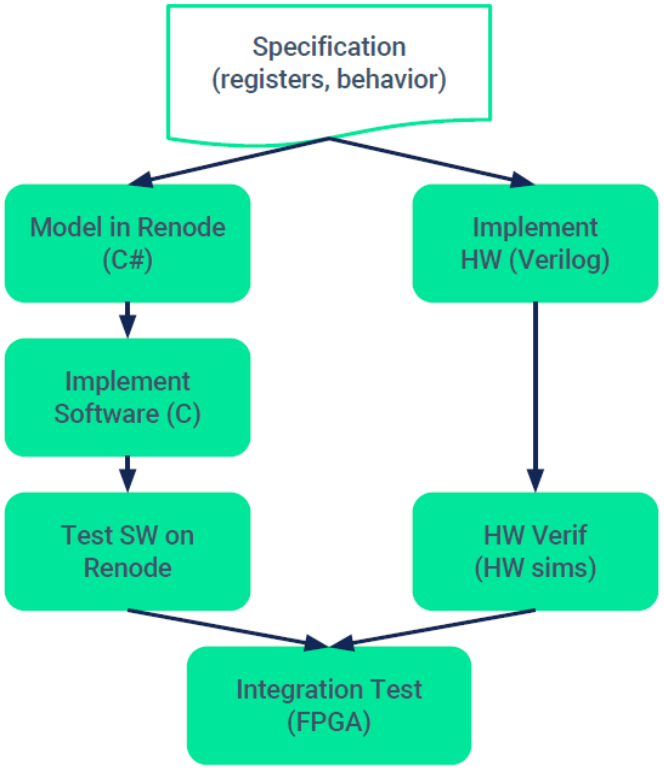
6) Parallel HW/SW Development

6.1 Overview

- 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.



Source: "Hardware/Software Co-Design with the Open Source Renode Framework and RISC-V", Michael Gielda, Getting Started With RISC-V NA Tour 2019.

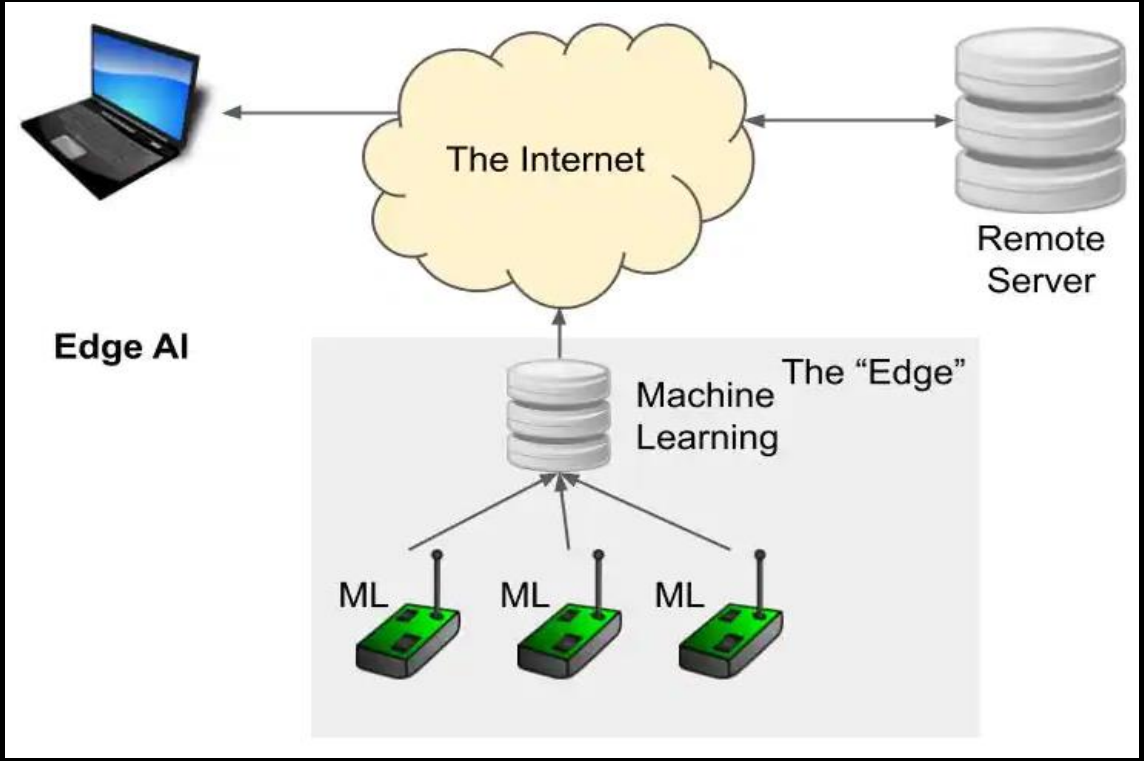
- …

IV. Renode for Edge AI

2021.NET Conf China
开源共建 | 开放创新 | 开发赋能

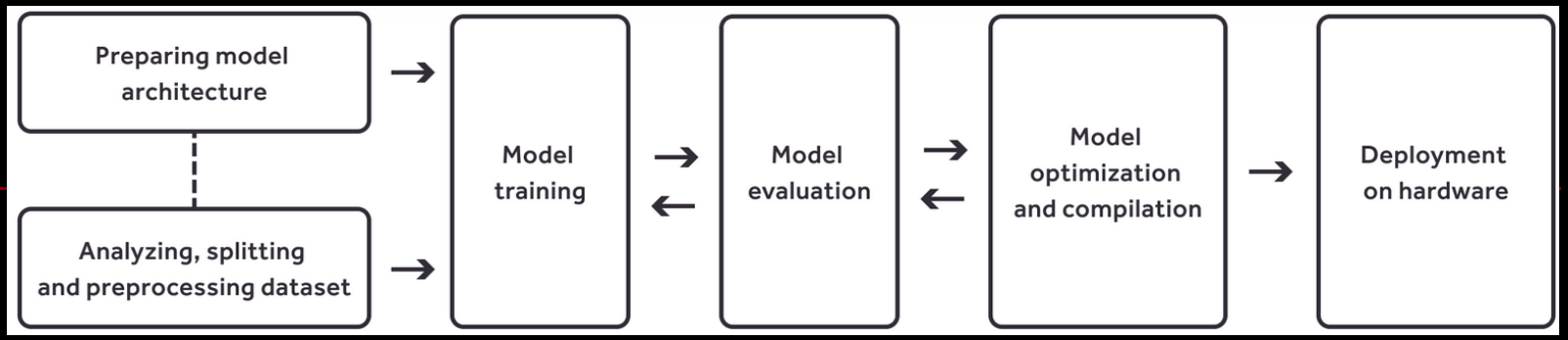
1) Overview

- https://www.digikey.com/en/maker/projects/what-is-edge-ai-machine-learning-iot/4f65583813894113 8aaad62c170827af



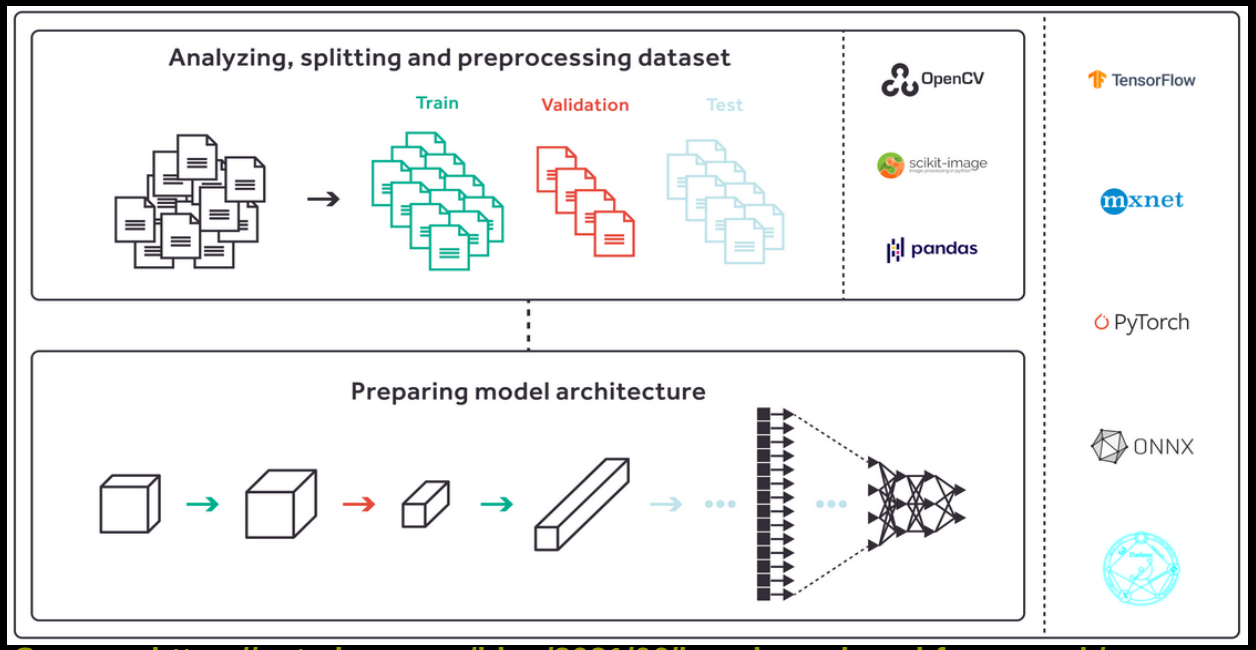
- https://antmicro.com/services/edge-ai/
- …

Pipeline



Source: https://antmicro.com/blog/2021/06/kenning-edge-ai-framework/

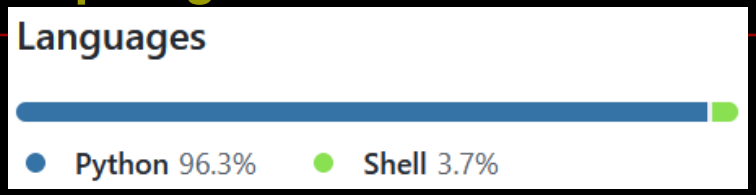
Edge AI deployment stack



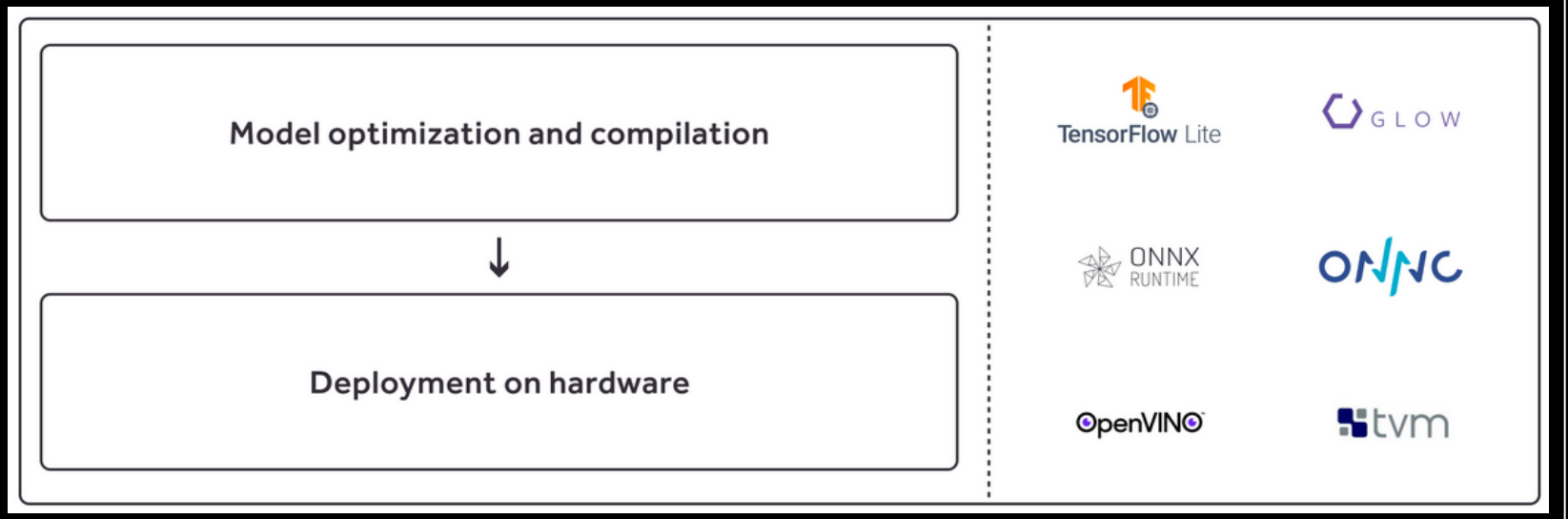
Source: https://antmicro.com/blog/2021/06/kenning-edge-ai-framework/

Kenning Framework

- https://antmicro.github.io/kenning/
  Deploying deep learning models on the edge.
- https://github.com/antmicro/kenning/

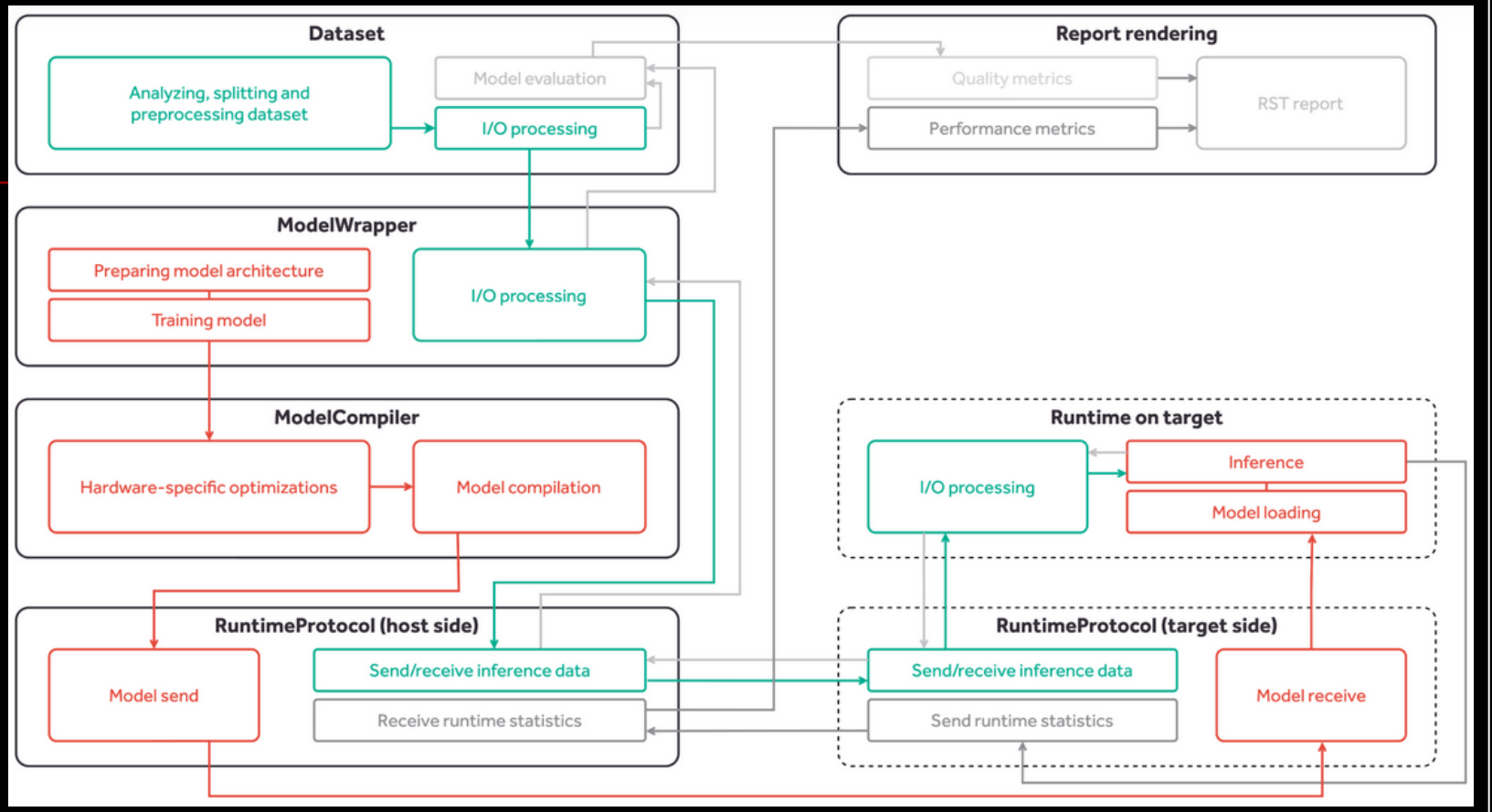


- https://antmicro.com/blog/2021/06/kenning-edge-ai-framework/



- https://antmicro.com/blog/2021/11/kenning-runtime/
- …

Workflow and Classes



Source: https://antmicro.com/blog/2021/06/kenning-edge-ai-framework/

2) VEDLIoT

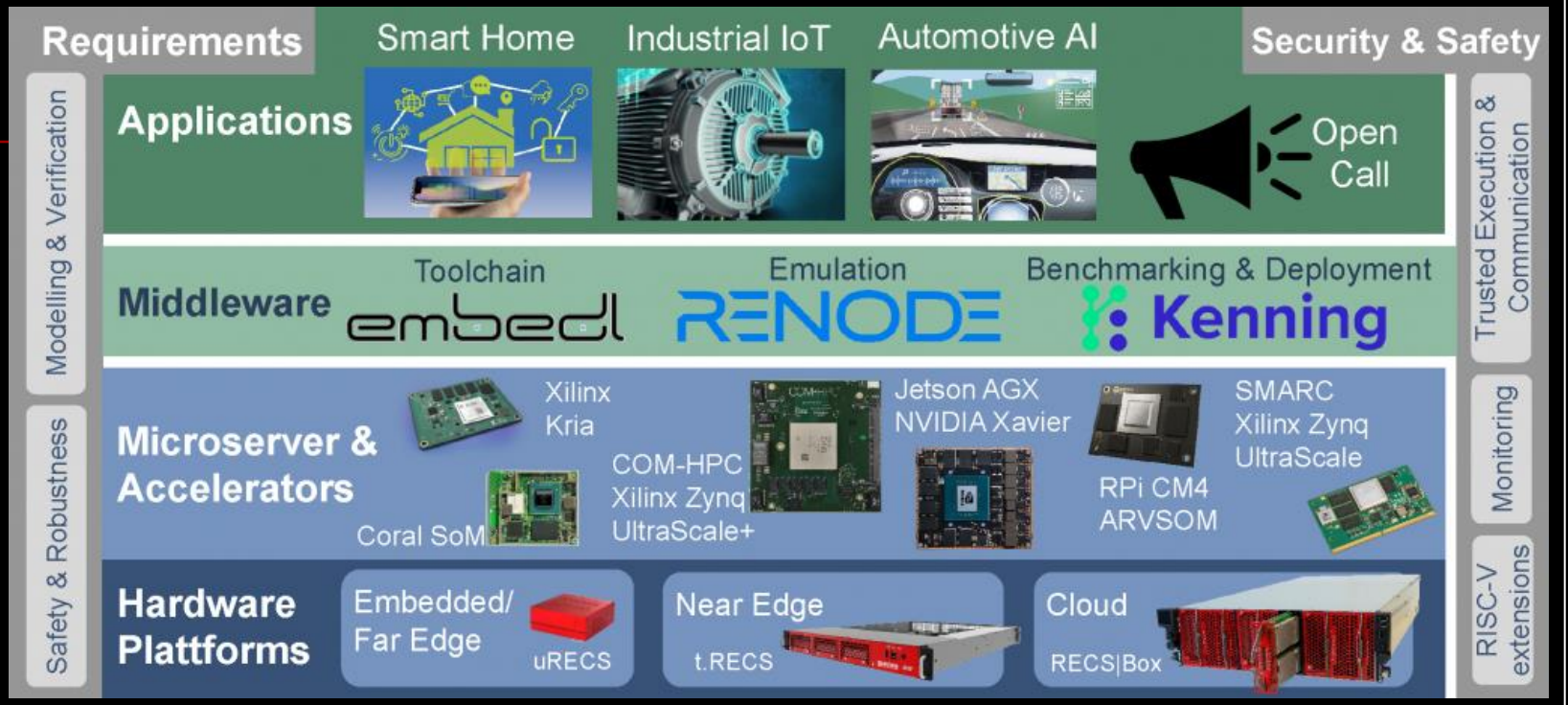
- https://vedliot.eu/
  Very Efficient Deep Learning in IoT project.
- https://vedliot.eu/project-summary/
- https://antmicro.com/blog/2021/06/kenning-edge-ai-framework/

> Kenning's origin lies in a benchmarking framework implemented during our work on Very Efficient Deep Learning in IoT project. VEDLIoT is focused on building platform-independent AI for edge applications and exploring RISC-V (mentioned earlier in our blog in Very Efficient Deep Learning in IoT project with RISC-V and Renode.) That's another instance of our constant efforts on many fronts towards more open AI solutions in general, aligning with our work around open source FPGA tooling, TF Lite Micro, Renode, CFU extensions for RISC-V and with other global leaders at CHIPS Alliance.

- https://vedliot.eu/hardware/
- https://vedliot.eu/toolchain/
- …

Structure

- https://vedliot.eu/vedliot-paper-at-date-2022/



3) Renode can help

3.1 Overview

- Our tools empower organizations like Google, Arm, QuickLogic, Microchip with test-driven open source methodologies of developing TinyML
- Google and Arm use Renode to develop and test TensorFlow Lite
- QuickLogic embraced open FPGA tools, Renode and other open technologies from Antmicro
- Microchip hired us to develop their pre-silicon development platform for RISC-V based PolarFire SoC with Renode

Source: "Developing and testing TensorFlow Lite Micro edge AI algorithms on RISC-V and FPGAs", Michael Gielda, London Open Source Meetup for RISC-V 2021.

- …

3.2 TinyML

3.2.1 TensorFlow Lite

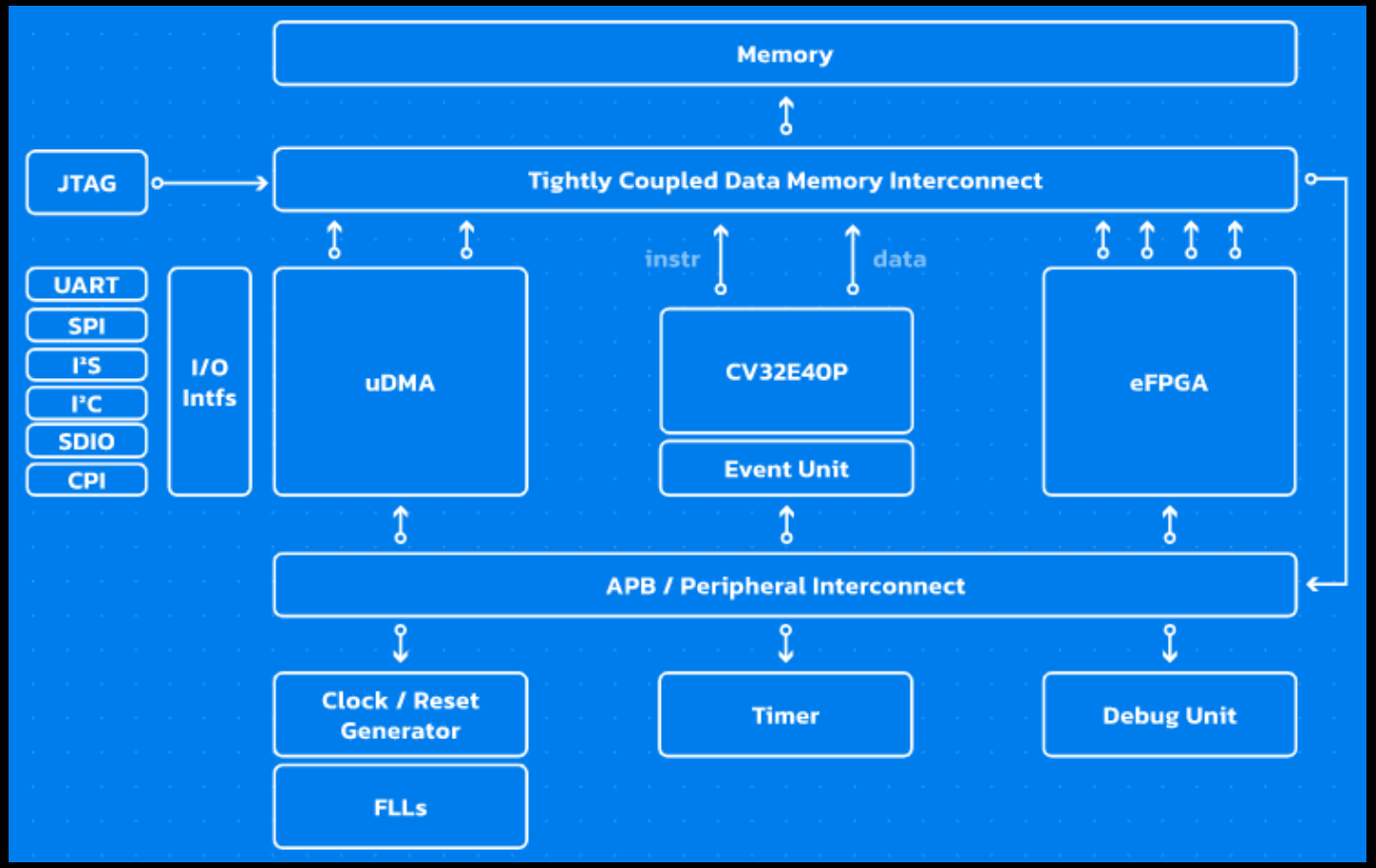
- https://antmicro.com/blog/2020/06/tf-lite-in-renode/
- https://github.com/antmicro/litex-vexriscv-tensorflow-lite-demo
- RENODE + TF LITE IN 5 MINUTES: GOOGLE COLAB

  - A (mostly) Python workspace in the cloud
  - Built with Jupyter notebooks
  - Allows you to run arbitrary scripts and share the results online
  - Fantastic tool for presentation purposes
  - Very popular among students and academics
  - Together with Google we implemented colabs which can process video and audio recorded from your computer
  - To be used in EdX course with over 16000 participants
  - See example colab

  Source: "Developing and testing TensorFlow Lite Micro edge AI algorithms on RISC-V and FPGAs", Michael Gielda, London Open Source Meetup for RISC-V 2021.
- https://antmicro.com/blog/2020/12/open-source-fpga-tools-and-renode-for-core-v/

3.2.2 Core-V

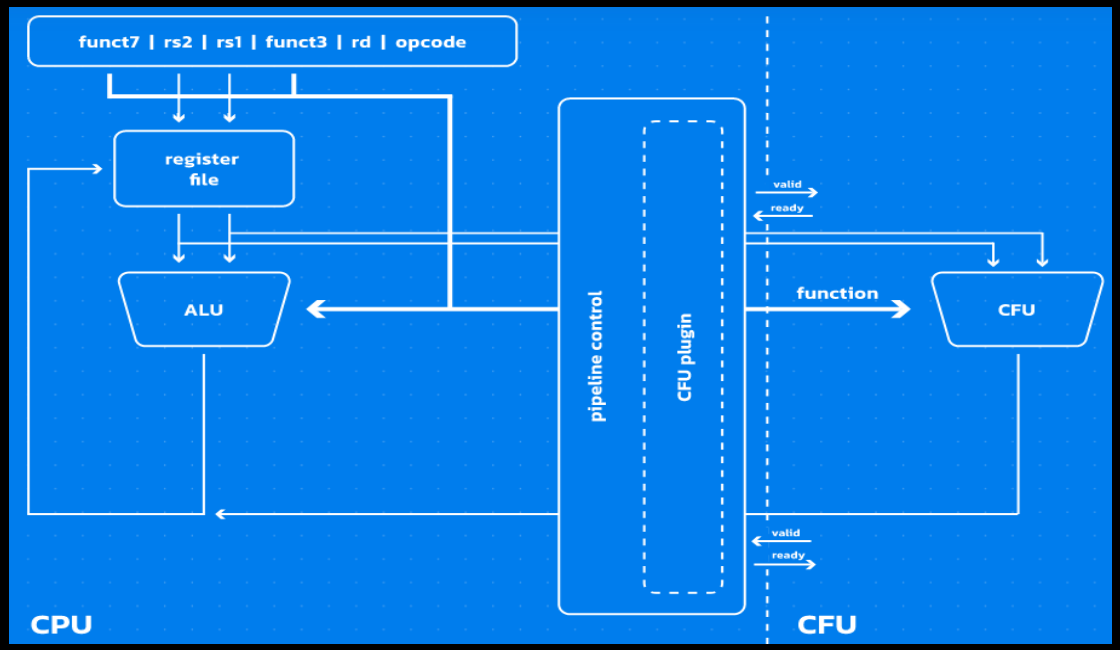
- https://docs.openhwgroup.org/projects/core-v-mcu/
- https://github.com/openhwgroup/core-v-mcu
- https://antmicro.com/blog/2020/12/open-source-fpga-tools-and-renode-for-core-v/



3.3 CFU

- Custom Function Units
- https://antmicro.com/blog/2021/09/cfu-support-in-renode/

RISC-V is also excellent for FPGA-based ML development, offering a multitude of FPGA-friendly softcore options such as VexRiscv and specialized ML-oriented extensions called CFU - which you can experiment with both in cheap, easily accessible hardware as well as, you guessed it, with Renode, using Verilator co-simulation capabilities that we have described a few times already.



3.3.1 CFU Playground

- https://cfu-playground.readthedocs.io/en/latest/

Accelerate ML models on FPGAs.

"CFU" stands for Custom Function Unit: accelerator hardware that is tightly coupled into the pipeline of a CPU core, to add new custom function instructions that complement the CPU's standard functions (such as arithmetic/logic operations).

The CFU Playground is a collection of software, gateware and hardware configured to make it easy to:

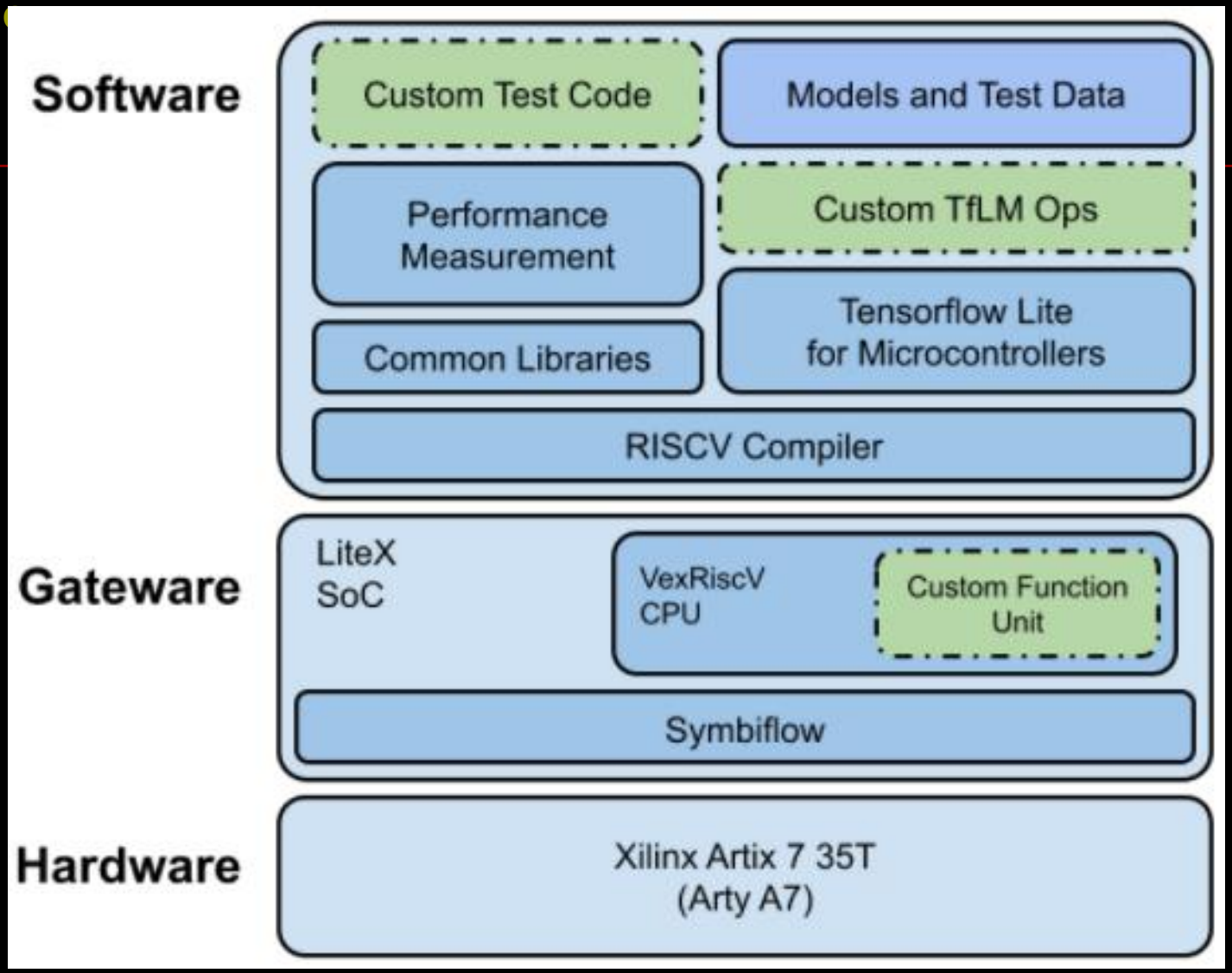
- Run ML models
- Benchmark and profile performance
- Make incremental improvements
  - In software by modifying source code
  - In gateware with a CFU
- Measure the results of changes

ML acceleration on microcontroller-class hardware is a new area, and one that, due to the expense of building ASICs, is currently dominated by hardware engineers. In order to encourage software engineers to join in the innovation, the CFU-Playground aims to make experimentation as simple, fast and fun as possible.

As well as being a useful tool for accelerating ML inferencing, the CFU Playground is a relatively gentle introduction to using FPGAs for computation.

- A new kind of ASIP?
- …

Architecture & Design



3.4 RISC-V vector instructions support

- https://github.com/riscv/riscv-v-spec/
- https://antmicro.com/blog/2021/12/riscv-vector-instructions-in-renode/

V. Renode on RPi4B

- Installing dependencies

Mono/.NET

Renode requires Mono >= 5.20 (Linux, macOS) or .NET >= 4.7 (Windows).

Linux	Install the `mono-complete` package as per the installation instructions for various Linux distributions which can be found on the Mono project website.
macOS	On macOS, the Mono package can be downloaded directly from the Mono project website.
Windows	On Windows 7, download and install .NET Framework 4.7. Windows 10 ships with .NET by default, so no action is required there.

1) Porting

Workload for AArch64

- reconstruct the build system
- prepare the prebuilt resource files for AArch64
- porting tlib to AArch64
- …

Heavy work…
Working on it, and hope to make exciting progress in next year…

2) A special prebuilt version

-

renode-1.11.0-1-x86_64.pkg.tar.xz
renode-1.11.0-1.f23.x86_64.rpm
renode-1.11.0.linux-portable.tar.gz
renode_1.11.0.dmg
renode_1.11.0.msi
renode_1.11.0.zip
renode_1.11.0_amd64.deb
Source code (zip)
Source code (tar.gz)

Mono/.Net assembly only

Running it on Fedora/Manjaro AArch64 for RPi4B(2021H1)

```
[mydev@MyRPi4Manjaro2 bin]$ mono ./Renode.exe --port 55555
12:02:05.0007 [WARNING] Couldn't start UI - falling back to telnet mode
12:02:06.1885 [INFO] Loaded monitor commands from: /opt/MyWorkSpace/DevSW/Sim-Modeling/Renode/r1.11.0/./scripts/monitor.py
12:02:06.2750 [INFO] Monitor available in telnet mode on port 55555
12:05:28.9137 [INFO] System bus created.
Fatal error:
```

...

```
[ERROR] FATAL UNHANDLED EXCEPTION: System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. ---> S
stem.DllNotFoundException: kernel32 assembly:<unknown assembly> type:<unknown type> member:(null)
  at (wrapper managed-to-native) Antmicro.Renode.Utilities.SharedLibraries.WindowsLoadLibrary(string)
  at Antmicro.Renode.Utilities.SharedLibraries.TryLoadLibrary (System.String path, System.IntPtr& address) [0x00000] in <c0b10643a8564222a33e17b
45d53209>:0
  at Antmicro.Renode.Utilities.SharedLibraries.LoadLibrary (System.String path) [0x00000] in <c0b10643a8564222a33e17b845d53209>:0
  at Antmicro.Renode.Utilities.Binding.NativeBinder..ctor (Antmicro.Renode.IEmulationElement classToBind, System.String libraryFile) [0x00019] i
 <c0b10643a8564222a33e17b845d53209>:0
  at Antmicro.Renode.Peripherals.CPU.TranslationCPU.Init () [0x000a4] in <ccd0ece1ea6442a093ef0525a57331c5>:0
  at Antmicro.Renode.Peripherals.CPU.TranslationCPU..ctor (System.UInt32 id, System.String cpuType, Antmicro.Renode.Core.Machine machine, ELFSha
p.ELF.Endianess endianness, Antmicro.Renode.Peripherals.CPU.CpuBitness bitness) [0x000de] in <ccd0ece1ea6442a093ef0525a57331c5>:0
  at Antmicro.Renode.Peripherals.CPU.BaseRiscV..ctor (Antmicro.Renode.Peripherals.Timers.IRiscVTimeProvider timeProvider, System.UInt32 hartId,
ystem.String cpuType, Antmicro.Renode.Core.Machine machine, Antmicro.Renode.Peripherals.CPU.BaseRiscV+PrivilegeArchitecture privilegeArchitectur
, ELFSharp.ELF.Endianess endianness, Antmicro.Renode.Peripherals.CPU.CpuBitness bitness, System.Nullable`1[T] nmiVectorAddress, System.Nullable`
[T] nmiVectorLength) [0x0012c] in <38bea6321ff24c429448b1e05f8339cf>:0
  at Antmicro.Renode.Peripherals.CPU.RiscV32..ctor (Antmicro.Renode.Peripherals.Timers.IRiscVTimeProvider timeProvider, System.String cpuType, A
tmicro.Renode.Core.Machine machine, System.UInt32 hartId, Antmicro.Renode.Peripherals.CPU.BaseRiscV+PrivilegeArchitecture privilegeArchitecture,
ELFSharp.ELF.Endianess endianness, System.Nullable`1[T] nmiVectorAddress, System.Nullable`1[T] nmiVectorLength) [0x00000] in <38bea6321ff24c4294
8b1e05f8339cf>:0
  at Antmicro.Renode.Peripherals.CPU.VexRiscv..ctor (Antmicro.Renode.Core.Machine machine, System.UInt32 hartId, Antmicro.Renode.Peripherals.Tim
rs.IRiscVTimeProvider timeProvider, Antmicro.Renode.Peripherals.CPU.BaseRiscV+PrivilegeArchitecture privilegeArchitecture, System.String cpuType
 [0x0000b] in <38bea6321ff24c429448b1e05f8339cf>:0
  at (wrapper managed-to-native) System.Reflection.RuntimeConstructorInfo.InternalInvoke(System.Reflection.RuntimeConstructorInfo,object,object[
,System.Exception&)
  at System.Reflection.RuntimeConstructorInfo.InternalInvoke (System.Object obj, System.Object[] parameters, System.Boolean wrapExceptions) [0x0
005] in <efe941bb62534dc3a62ceb1a818964a0>:0
   --- End of inner exception stack trace ---
```

```
[mydev@MyRPi4Manjaro2 ~]$ telnet 127.0.0.1 55555
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Renode, version 1.11.0.28399 (ad7ab75a-202010221522)

(monitor) mach create "litex-vexriscv-linux"
(litex-vexriscv-linux) machine LoadPlatformDescription @platforms/cpus/litex_vexriscv_linux.repl
Connection closed by foreign host.
[mydev@MyRPi4Manjaro2 ~]$
```

- This special version of Renode can be successfully launched (only in headless mode due to fail to start the UI, but we can still connect to Renode Monitor via the specified port), then a runtime error happened when we tried to create a simulated instance of litex-vexriscv-linux:
  The DllNotFoundException that thrown out -- failed to load the corresponding DLL.

Running it on Ubuntu AArch32 for RPi4B(2021H1)


```
ubuntu@ubuntu:/opt/MyWorkSpace/DevSW/Sim-Modeling/Renode/r1.11.0/bin$ mono ./Renode.exe --port 55555

=================================================================
        Native Crash Reporting
=================================================================
Got a SIGSEGV while executing native code. This usually indicates
a fatal error in the mono runtime or one of the native libraries
used by your application.
=================================================================

=================================================================
        Native stacktrace:
=================================================================
         (No frames)
```

```
=================================================================
        Basic Fault Address Reporting
=================================================================
Memory around native instruction pointer (0xb6e65584):0xb6e65574  01 46 9c e7 c6 4f 00 00 2d e9 f0 4f 85 b0 00 bf  .F...O..-..O....
0xb6e65584  47 6a fb 08 00 f0 b7 81 04 46 17 f0 01 07 05 46  Gj.......F.....F
0xb6e65594  18 bf 80 27 54 f8 20 3f 0e e0 43 f0 01 01 54 e8  ...'T. ?..C...T.
0xb6e655a4  00 2f 9a 42 05 d1 44 e8 00 10 00 28 f7 d1 bf f3  ./.B..D....(....

=================================================================
        Managed Stacktrace:
=================================================================
          at <unknown> <0xffffffff>
          at Antmicro.Renode.Peripherals.BackendManager:Init <0x00137>
          at Antmicro.Renode.Peripherals.BackendManager:.ctor <0x0005b>
          at Antmicro.Renode.Core.Emulation:.ctor <0x0060f>
          at Antmicro.Renode.Core.EmulationManager:.ctor <0x0038f>
          at Antmicro.Renode.Core.EmulationManager:RebuildInstance <0x00037>
          at Antmicro.Renode.Core.EmulationManager:.cctor <0x0001f>
          at System.Object:runtime_invoke_void <0x0007b>
          at <unknown> <0xffffffff>
          at Antmicro.Renode.Logging.Logger:LogAs <0x0001f>
          at Antmicro.Renode.Logging.Logger:Log <0x0002b>
          at Antmicro.Renode.Logging.Logger:Log <0x00027>
          at Antmicro.Renode.UI.CommandLineInterface:Run <0x00463>
          at <>c__DisplayClass0_0:<Main>b__1 <0x00163>
          at System.Threading.ThreadHelper:ThreadStart_Context <0x000df>
          at System.Threading.ExecutionContext:RunInternal <0x00203>
          at System.Threading.ExecutionContext:Run <0x0002b>
          at System.Threading.ExecutionContext:Run <0x00067>
          at System.Threading.ThreadHelper:ThreadStart <0x00047>
          at System.Object:runtime_invoke_void__this__ <0x000eb>
=================================================================
Aborted (core dumped)
```

- **Suspect this failure is caused by Mono, rebuild it from source (https://github.com/mono/mono) for retesting:**

```
ubuntu@ubuntu:/opt/MyWorkSpace/DevSW/Sim-Modeling/Renode/r1.11.0/bin$ /opt/MyWorkSpace/MyProjs/Runtime/DotNet/Mono/mono-main/build/Install/bin/mo
no ./Renode.exe --port 55555
15:05:39.7592 [WARNING] Couldn't start UI - falling back to telnet mode
15:05:42.7343 [INFO] Loaded monitor commands from: /opt/MyWorkSpace/DevSW/Sim-Modeling/Renode/r1.11.0/./scripts/monitor.py
15:05:42.9878 [INFO] Monitor available in telnet mode on port 55555
15:11:59.6687 [INFO] System bus created.
Fatal error:
Exception has been thrown by the target of an invocation.
  at System.Reflection.RuntimeConstructorInfo.InternalInvoke (System.Object obj, System.Object[] parameters, System.Boolean wrapExceptions) [0x00
01a] in <78e2ffd775544956940f098edf0c4908>:0
  at System.Reflection.RuntimeConstructorInfo.DoInvoke (System.Object obj, System.Reflection.BindingFlags invokeAttr, System.Reflection.Binder bi
nder, System.Object[] parameters, System.Globalization.CultureInfo culture) [0x00086] in <78e2ffd775544956940f098edf0c4908>:0
  at System.Reflection.RuntimeConstructorInfo.Invoke (System.Reflection.BindingFlags invokeAttr, System.Reflection.Binder binder, System.Object[]
 parameters, System.Globalization.CultureInfo culture) [0x00000] in <78e2ffd775544956940f098edf0c4908>:0
  at System.Reflection.ConstructorInfo.Invoke (System.Object[] parameters) [0x00000] in <78e2ffd775544956940f098edf0c4908>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.CreateAndHandleError (System.Reflection.ConstructorInfo constructor, System.Collections.G
eneric.IEnumerable`1[T] attributes, System.String friendlyName, Antmicro.Renode.PlatformDescription.Syntax.IWithPosition responsibleSyntaxElement
) [0x0004a] in <5e6277b90b194ccab98fac7a84ae0435>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.CreateFromEntry (Antmicro.Renode.PlatformDescription.Syntax.Entry entry) [0x00034] in <5e
6277b90b194ccab98fac7a84ae0435>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.ProcessInner (System.String file, System.String source) [0x00061] in <5e6277b90b194ccab98
fac7a84ae0435>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.ProcessFile (System.String path) [0x00031] in <5e6277b90b194ccab98fac7a84ae0435>:0
  at Antmicro.Renode.PlatformDescription.UserInterface.PlatformDescriptionMachineExtensions.LoadPlatformDescription (Antmicro.Renode.Core.Machine
 machine, System.String platformDescriptionFile) [0x00006] in <5e6277b90b194ccab98fac7a84ae0435>:0
  at (wrapper dynamic-method) System.Object.CallSite.Target(System.Runtime.CompilerServices.Closure,System.Runtime.CompilerServices.CallSite,obje
ct,object,object)
  at System.Dynamic.UpdateDelegates.UpdateAndExecuteVoid3[T0,T1,T2] (System.Runtime.CompilerServices.CallSite site, T0 arg0, T1 arg1, T2 arg2) [0
x0011e] in <5e4c6cf5748a4ac5aee94bbd277a3508>:0
  at (wrapper delegate-invoke) System.Action`4[System.Runtime.CompilerServices.CallSite,System.Object,System.Object,System.Object].invoke_void_T1
_T2_T3_T4(System.Runtime.CompilerServices.CallSite,object,object,object)
  at Dynamitey.Internal.Optimization.InvokeHelper.InvokeMemberAction (System.Runtime.CompilerServices.CallSite& callsite, System.Type binderType,
 System.Int32 knownType, Dynamitey.Internal.Optimization.InvokeHelper+LazyBinder binder, Dynamitey.InvokeMemberName name, System.Boolean staticCo
ntext, System.Type context, System.String[] argNames, System.Object target, System.Object[] args) [0x000df] in <1d0e1ef63ad2470a9ccec81105ee1302>
:0
  at Dynamitey.Internal.Optimization.InvokeHelper.InvokeMemberActionCallSite (System.Object target, Dynamitey.InvokeMemberName name, System.Objec
t[] args, System.String[] tArgNames, System.Type tContext, System.Boolean tStaticContext, System.Runtime.CompilerServices.CallSite& callSite) [0x
0004f] in <1d0e1ef63ad2470a9ccec81105ee1302>:0
  at Dynamitey.Dynamic.InvokeMemberAction (System.Object target, Dynamitey.String_OR_InvokeMemberName name, System.Object[] args) [0x00018] in <1
d0e1ef63ad2470a9ccec81105ee1302>:0
  at Antmicro.Renode.UserInterface.Monitor.InvokeWithContext (Dynamitey.InvokeContext context, System.Reflection.MethodInfo method, System.Object
```

```
[ERROR] FATAL UNHANDLED EXCEPTION: System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. ---> Sy
stem.DllNotFoundException: kernel32 assembly:<unknown assembly> type:<unknown type> member:(null)
  at (wrapper managed-to-native) Antmicro.Renode.Utilities.SharedLibraries.WindowsLoadLibrary(string)
  at Antmicro.Renode.Utilities.SharedLibraries.TryLoadLibrary (System.String path, System.IntPtr& address) [0x00000] in <c0b10643a8564222a33e17b8
45d53209>:0
  at Antmicro.Renode.Utilities.SharedLibraries.LoadLibrary (System.String path) [0x00000] in <c0b10643a8564222a33e17b845d53209>:0
  at Antmicro.Renode.Utilities.Binding.NativeBinder..ctor (Antmicro.Renode.IEmulationElement classToBind, System.String libraryFile) [0x00019] in
 <c0b10643a8564222a33e17b845d53209>:0
  at Antmicro.Renode.Peripherals.CPU.TranslationCPU.Init () [0x000a4] in <ccd0ece1ea6442a093ef0525a57331c5>:0
  at Antmicro.Renode.Peripherals.CPU.TranslationCPU..ctor (System.UInt32 id, System.String cpuType, Antmicro.Renode.Core.Machine machine, ELFShar
p.ELF.Endianess endianness, Antmicro.Renode.Peripherals.CPU.CpuBitness bitness) [0x000de] in <ccd0ece1ea6442a093ef0525a57331c5>:0
  at Antmicro.Renode.Peripherals.CPU.BaseRiscV..ctor (Antmicro.Renode.Peripherals.Timers.IRiscVTimeProvider timeProvider, System.UInt32 hartId, S
ystem.String cpuType, Antmicro.Renode.Core.Machine machine, Antmicro.Renode.Peripherals.CPU.BaseRiscV+PrivilegeArchitecture privilegeArchitecture
, ELFSharp.ELF.Endianess endianness, Antmicro.Renode.Peripherals.CPU.CpuBitness bitness, System.Nullable`1[T] nmiVectorAddress, System.Nullable`1
[T] nmiVectorLength) [0x0012c] in <38bea6321ff24c429448b1e05f8339cf>:0
  at Antmicro.Renode.Peripherals.CPU.RiscV32..ctor (Antmicro.Renode.Peripherals.Timers.IRiscVTimeProvider timeProvider, System.String cpuType, An
tmicro.Renode.Core.Machine machine, System.UInt32 hartId, Antmicro.Renode.Peripherals.CPU.BaseRiscV+PrivilegeArchitecture privilegeArchitecture,
ELFSharp.ELF.Endianess endianness, System.Nullable`1[T] nmiVectorAddress, System.Nullable`1[T] nmiVectorLength) [0x00000] in <38bea6321ff24c42944
8b1e05f8339cf>:0
  at Antmicro.Renode.Peripherals.CPU.VexRiscv..ctor (Antmicro.Renode.Core.Machine machine, System.UInt32 hartId, Antmicro.Renode.Peripherals.Time
rs.IRiscVTimeProvider timeProvider, Antmicro.Renode.Peripherals.CPU.BaseRiscV+PrivilegeArchitecture privilegeArchitecture, System.String cpuType)
 [0x0000b] in <38bea6321ff24c429448b1e05f8339cf>:0
  at (wrapper managed-to-native) System.Reflection.RuntimeConstructorInfo.InternalInvoke(System.Reflection.RuntimeConstructorInfo,object,object[]
,System.Exception&)
  at System.Reflection.RuntimeConstructorInfo.InternalInvoke (System.Object obj, System.Object[] parameters, System.Boolean wrapExceptions) [0x00
005] in <78e2ffd775544956940f098edf0c4908>:0
   --- End of inner exception stack trace ---
  at System.Reflection.RuntimeConstructorInfo.InternalInvoke (System.Object obj, System.Object[] parameters, System.Boolean wrapExceptions) [0x00
01a] in <78e2ffd775544956940f098edf0c4908>:0
  at System.Reflection.RuntimeConstructorInfo.DoInvoke (System.Object obj, System.Reflection.BindingFlags invokeAttr, System.Reflection.Binder bi
nder, System.Object[] parameters, System.Globalization.CultureInfo culture) [0x00086] in <78e2ffd775544956940f098edf0c4908>:0
  at System.Reflection.RuntimeConstructorInfo.Invoke (System.Reflection.BindingFlags invokeAttr, System.Reflection.Binder binder, System.Object[]
 parameters, System.Globalization.CultureInfo culture) [0x00000] in <78e2ffd775544956940f098edf0c4908>:0
  at System.Reflection.ConstructorInfo.Invoke (System.Object[] parameters) [0x00000] in <78e2ffd775544956940f098edf0c4908>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.CreateAndHandleError (System.Reflection.ConstructorInfo constructor, System.Collections.G
eneric.IEnumerable`1[T] attributes, System.String friendlyName, Antmicro.Renode.PlatformDescription.Syntax.IWithPosition responsibleSyntaxElement
) [0x0004a] in <5e6277b90b194ccab98fac7a84ae0435>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.CreateFromEntry (Antmicro.Renode.PlatformDescription.Syntax.Entry entry) [0x00034] in <5e
6277b90b194ccab98fac7a84ae0435>:0
  at Antmicro.Renode.PlatformDescription.CreationDriver.ProcessInner (System.String file, System.String source) [0x00061] in <5e6277b90b194ccab98
fac7a84ae0435>:0
```

```
ubuntu@ubuntu:/$ telnet 127.0.0.1 55555
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Renode, version 1.11.0.28399 (ad7ab75a-202010221522)

(monitor) mach create "litex-vexriscv-linux"
(litex-vexriscv-linux) machine LoadPlatformDescription @platforms/cpus/litex_vexriscv_linux.repl
Connection closed by foreign host.
ubuntu@ubuntu:/$
```

- This special version of Renode failed to launch on an AArch32 distribution directly, this seems to be something wrong with the Mono version installed on our Ubuntu AArch32, since the same Renode executable can be launched even on AArch64 as described above. So we decided to abandon the version of Mono(https://github.com/mono/mono) that installed through the package manager, and rebuilt it from the latest source code by our own on RPi4. This new version can successfully launch Renode, this result confirmed our previous suspicion, but finally it still met the same failure as that on AArch64 while loading shared library.

Running the latest build on Fedora34 AArch64(2021H2)

- https://dl.antmicro.com/projects/renode/builds/

renode-latest.deb	2021-12-16 09:03	30M
renode-latest.x86_64.rpm	2021-12-16 09:03	30M
renode-latest.linux-portable.tar.gz	2021-12-16 09:03	49M
renode-latest.pkg.tar.xz	2021-12-16 09:03	20M
renode_1.12.0+20211216gitfb16507c_amd64.deb	2021-12-16 09:03	30M
renode-1.12.0+20211216gitfb16507c-1.x86_64.rpm	2021-12-16 09:03	30M
renode-1.12.0+20211216gitfb16507c.linux-portable.tar.gz	2021-12-16 09:02	49M
renode-1.12.0+20211216gitfb16507c-1-x86_64.pkg.tar.xz	2021-12-16 09:02	20M
renode-latest.dmg	2021-12-16 09:02	34M
renode_1.12.0+20211216gitfb16507c.dmg	2021-12-16 09:02	34M
renode-latest.zip	2021-12-16 09:02	32M
renode-latest.msi	2021-12-16 09:02	21M
renode_1.12.0+20211216gitfb16507c.zip	2021-12-16 09:02	32M
renode_1.12.0+20211216gitfb16507c.msi	2021-12-16 09:02	21M

```
[mydev@fedora bin]$ uname -a
Linux fedora 5.15.7-100.fc34.aarch64 #1 SMP Wed Dec 8 19:00:11 UTC 2021 aarch64 aarch64 aarch64 GNU/Linux
[mydev@fedora bin]$
[mydev@fedora bin]$ dotnet --version
6.0.100
[mydev@fedora bin]$
[mydev@fedora bin]$ mono -V
Mono JIT compiler version 6.12.0.122 (tarball Sat Feb 27 17:39:10 UTC 2021)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-project.com
        TLS:           __thread
        SIGSEGV:       normal
        Notifications: epoll
        Architecture:  arm64
        Disabled:      none
        Misc:          softdebug
        Interpreter:   yes
        LLVM:          supported, not enabled.
        Suspend:       preemptive
        GC:            sgen (concurrent by default)
[mydev@fedora bin]$
```

the same failure as what in 2021H1

```
[mydev@fedora bin]$ mono ./Renode.exe --port 55555
19:08:41.6735 [WARNING] Couldn't start UI - falling back to telnet mode
19:08:43.3319 [INFO] Loaded monitor commands from: /opt/MyWorkSpace/MyProjs/Sim-Modeling/IoT-Edge/Renode/Releases/Latest/renod
e_1.12.0+20211216gitfb16507c/scripts/monitor.py
19:08:43.4137 [INFO] Monitor available in telnet mode on port 55555
19:26:37.7016 [INFO] System bus created.
Fatal error:
Exception has been thrown by the target of an invocation.
  at System.Reflection.RuntimeConstructorInfo.InternalInvoke (System.Object obj, System.Object[] parameters, System.Boolean wr
apExceptions) [0x00022] in <232d2ec5aa2f429a82060c0ba99debd7>:0
```

...

```
[ERROR] FATAL UNHANDLED EXCEPTION: System.Reflection.TargetInvocationException: Exception has been thrown by the target of an
invocation. ---> System.DllNotFoundException: kernel32 assembly:<unknown assembly> type:<unknown type> member:(null)
  at (wrapper managed-to-native) Antmicro.Renode.Utilities.SharedLibraries.WindowsLoadLibrary(string)
  at Antmicro.Renode.Utilities.SharedLibraries.TryLoadLibrary (System.String path, System.IntPtr& address) [0x00000] in <d1abb
ed5eac3446484c12b109b834ff5>:0
  at Antmicro.Renode.Utilities.SharedLibraries.LoadLibrary (System.String path) [0x00000] in <d1abbed5eac3446484c12b109b834ff5
>:0
  at Antmicro.Renode.Utilities.Binding.NativeBinder..ctor (Antmicro.Renode.IEmulationElement classToBind, System.String librar
yFile) [0x00019] in <d1abbed5eac3446484c12b109b834ff5>:0
  at Antmicro.Renode.Peripherals.CPU.TranslationCPU.Init () [0x000a4] in <52ff4a7595f445da8e23bc112b3a6e28>:0
  at Antmicro.Renode.Peripherals.CPU.TranslationCPU..ctor (System.UInt32 id, System.String cpuType, Antmicro.Renode.Core.Machi
ne machine, ELFSharp.ELF.Endianess endianness, Antmicro.Renode.Peripherals.CPU.CpuBitness bitness) [0x000e2] in <52ff4a7595f44
5da8e23bc112b3a6e28>:0
```

...

```
[mydev@fedora /]$ telnet 127.0.0.1 55555
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Renode, version 1.12.0.6092 (fb16507c-202112160322)

(monitor) mach create "litex-vexriscv-linux"
(litex-vexriscv-linux)
(litex-vexriscv-linux) machine LoadPlatformDescription @platforms/cpus/litex_vexriscv_linux.repl
Connection closed by foreign host.
[mydev@fedora /]$
```

retried with export MONO_PATH

```
[mydev@fedora bin]$ ls -l /usr/lib64/wine/aarch64-windows/kernel32.dll
-rw-r--r--. 1 root root 1812576 Nov 11 06:17 /usr/lib64/wine/aarch64-windows/kernel32.dll
[mydev@fedora bin]$ file /usr/lib64/wine/aarch64-windows/kernel32.dll
/usr/lib64/wine/aarch64-windows/kernel32.dll: PE32+ executable (DLL) Aarch64, for MS Windows
[mydev@fedora bin]$
```

or /home/mydev/.wine/drive_c/windows/system32/kernel32.dll

```
[mydev@fedora bin]$ export MONO_PATH=/usr/lib64/wine/aarch64-windows
```

…

still failed with the same DllNotFoundException

retried with GAC

https://www.mono-project.com/docs/advanced/assemblies-and-the-gac/

```
[mydev@fedora /]$ pushd /usr/lib64/wine/aarch64-windows
/usr/lib64/wine/aarch64-windows /
[mydev@fedora aarch64-windows]$ gacutil -i kernel32.dll
Failure adding assembly kernel32.dll to the cache: The file specified is not a valid assembly.
[mydev@fedora aarch64-windows]$
```

…

https://stackoverflow.com/questions/61405038/using-extern-functions-depending-on-kernel32-dll-with-mono
https://mono-list.ximian.narkive.com/MwgdcYUj/can-we-add-kernel32-dll-in-mono

- Retrying to build this special Renode version against Linux or retried it on Windows for RPi4…

Run with the dotnet

```
[mydev@fedora bin]$ cat Renode.runtimeconfig.json
{
  "sdk": {
    "version": "6.0.100",
    "rollForward": "latestFeature"
  }
}
[mydev@fedora bin]$
[mydev@fedora bin]$ dotnet exec ./Renode.exe --port 55555
Invalid runtimeconfig.json [/opt/MyWorkSpace/MyProjs/Sim-Modeling/IoT-Edge/Renode/Releases/Latest/renode_1.12.0+20211216gitfb1
6507c/bin/Renode.runtimeconfig.json] [/opt/MyWorkSpace/MyProjs/Sim-Modeling/IoT-Edge/Renode/Releases/Latest/renode_1.12.0+2021
1216gitfb16507c/bin/Renode.runtimeconfig.dev.json]
[mydev@fedora bin]$
```

3) Virtualization

3.1 Official Docker image

- https://github.com/renode/renode-docker
- Build the image

```
[mydev@fedora renode-docker-master]$ podman build -t renode_min_test ./ -f Dockerfile.min
[1/2] STEP 1/6: FROM ubuntu:21.04 AS BUILDER
[1/2] STEP 2/6: ARG RENODE_DEST_BUILDER
--> Using cache 289814b28d94aea693ca55865f077dbb4bbdc46be8e88a0ca037829177e2215b
--> 289814b28d9
[1/2] STEP 3/6: ARG RENODE_URL
--> Using cache efcd30557473eafc2a8880bf2d5ab54decf91634359352420745a42b7de0b6fb
--> efcd3055747
[1/2] STEP 4/6: ARG RENODE_COMPRESSED_DIR=/opt/renode/compressed/
--> Using cache 705fb4a5b42e4b93acb3e219ae633c6793d67f19478df411782c6544f7b4de9a
--> 705fb4a5b42
[1/2] STEP 5/6: ADD $RENODE_URL $RENODE_COMPRESSED_DIR
[2/2] STEP 1/5: FROM mono:slim AS RUNTIME
Error: error building at STEP "ADD $RENODE_URL $RENODE_COMPRESSED_DIR": error reading "https://github.com/renode/renode/releases
/download/v1.12.0/renode-1.12.0.linux-portable.tar.gz": Get "https://github.com/renode/renode/releases/download/v1.12.0/renode-1
.12.0.linux-portable.tar.gz": unexpected EOF
[mydev@fedora renode-docker-master]$
[mydev@fedora renode-docker-master]$ podman build -t renode_min_test ./ -f Dockerfile.min
[1/2] STEP 1/6: FROM ubuntu:21.04 AS BUILDER
[1/2] STEP 2/6: ARG RENODE_DEST_BUILDER
--> Using cache 289814b28d94aea693ca55865f077dbb4bbdc46be8e88a0ca037829177e2215b
--> 289814b28d9
[1/2] STEP 3/6: ARG RENODE_URL
--> Using cache efcd30557473eafc2a8880bf2d5ab54decf91634359352420745a42b7de0b6fb
--> efcd3055747
[1/2] STEP 4/6: ARG RENODE_COMPRESSED_DIR=/opt/renode/compressed/
--> Using cache 705fb4a5b42e4b93acb3e219ae633c6793d67f19478df411782c6544f7b4de9a
--> 705fb4a5b42
[1/2] STEP 5/6: ADD $RENODE_URL $RENODE_COMPRESSED_DIR
--> Using cache 052ecb425b59f85882b8cb21bdc6d10b33467bb9a93059f173994ed702878f5f
--> 052ecb425b5
[1/2] STEP 6/6: RUN tar -vxf $RENODE_COMPRESSED_DIR/* -C $RENODE_DEST_BUILDER
--> Using cache c1821993a404bbda324049a1ab6244a7637a1694e820d0d04d4d985dc0e00def
--> c1821993a40
[2/2] STEP 1/5: FROM mono:slim AS RUNTIME
[2/2] STEP 2/5: ARG RENODE_DEST_BUILDER
--> e20c23e626f
[2/2] STEP 3/5: ARG RENODE_VERSION
--> 22749873186
[2/2] STEP 4/5: COPY --from=BUILDER ${RENODE_DEST_BUILDER}/renode_${RENODE_VERSION}_portable/ ${RENODE_DEST_BUILDER}
--> 19dd327dff4
[2/2] STEP 5/5: CMD ["/opt/renode/renode"]
[2/2] COMMIT renode_min_test
--> 7271ed3733f
Successfully tagged localhost/renode_min_test:latest
7271ed3733fda14d096f7b6b998534fd4d488024985aa3a492c3dc4cb8ed84bd
[mydev@fedora renode-docker-master]$
```

Run with the qemu-user-static

```
[mydev@fedora renode-docker-master]$ ls /proc/sys/fs/binfmt_misc/
qemu-aarch64_be  qemu-i486          qemu-mips64     qemu-or1k     qemu-riscv64  qemu-sparc32plus  register
qemu-alpha       qemu-m68k          qemu-mips64el   qemu-ppc      qemu-s390x    qemu-sparc64      status
qemu-armeb       qemu-microblaze    qemu-mipsel     qemu-ppc64    qemu-sh4      qemu-x86_64       windows
qemu-hppa        qemu-microblazeel  qemu-mipsn32    qemu-ppc64le  qemu-sh4eb    qemu-xtensa       windowsPE
qemu-i386        qemu-mips          qemu-mipsn32el  qemu-riscv32  qemu-sparc    qemu-xtensaeb
[mydev@fedora renode-docker-master]$
```

```
[mydev@fedora renode-docker-master]$ podman images
REPOSITORY                        TAG         IMAGE ID      CREATED        SIZE
localhost/renode_min_test         latest      7271ed3733fd  6 minutes ago  285 MB
<none>                            <none>      fd8d0f3b9136  10 hours ago   76.4 MB
<none>                            <none>      c1821993a404  10 hours ago   228 MB
docker.io/library/ubuntu          21.04       03acb1cbfc87  13 days ago    75 MB
docker.io/library/mono            slim        8985e0d0c74b  2 weeks ago    181 MB
docker.io/library/ubuntu          20.04       d5ca7a445605  2 months ago   68.1 MB
docker.io/library/alpine          latest      ae607a46d002  4 months ago   5.61 MB
registry.access.redhat.com/ubi8   latest      795e28a8b49a  4 months ago   267 MB
docker.io/antmicro/renode         latest      82caf80195b6  7 months ago   805 MB
docker.io/library/busybox         latest      d9d6c2bcb750  7 months ago   1.63 MB
[mydev@fedora renode-docker-master]$
[mydev@fedora renode-docker-master]$ podman run -p 55555:55555 localhost/renode_min_test
qemu-x86_64-static: Could not open '/lib64/ld-linux-x86-64.so.2': No such file or directory
[mydev@fedora renode-docker-master]$
```

checking the images…

```
[mydev@fedora renode-docker-master]$ podman run -p 55555:55555 antmicro/renode
TYPE: 3
V: AllocVector
* Assertion at ../../mono/arch/amd64/../x86/x86-codegen.h:410, condition `offset == (gint32)offset' not met


=================================================================
        Native Crash Reporting
=================================================================
Got a SIGABRT while executing native code. This usually indicates
a fatal error in the mono runtime or one of the native libraries
used by your application.
=================================================================

=================================================================
        Native stacktrace:
=================================================================
        0x40000bceb5 - /usr/bin/mono : (null)
        0x40000bd250 - /usr/bin/mono : (null)
        0x400006a129 - /usr/bin/mono : (null)
        0x40000bc467 - /usr/bin/mono : (null)
        0x4001e643c0 - /lib/x86_64-linux-gnu/libpthread.so.0 : (null)
        0x4001ed318b - /lib/x86_64-linux-gnu/libc.so.6 : gsignal
        0x4001eb2859 - /lib/x86_64-linux-gnu/libc.so.6 : abort
        0x400002d2eb - /usr/bin/mono : (null)
        0x400031a1a5 - /usr/bin/mono : (null)
        0x4000337b2e - /usr/bin/mono : (null)
        0x4000338113 - /usr/bin/mono : monoeg_assertion_message
        0x4000092e41 - /usr/bin/mono : (null)
        0x40000c16d3 - /usr/bin/mono : (null)
```

...

3.2 Run with a X64 VM

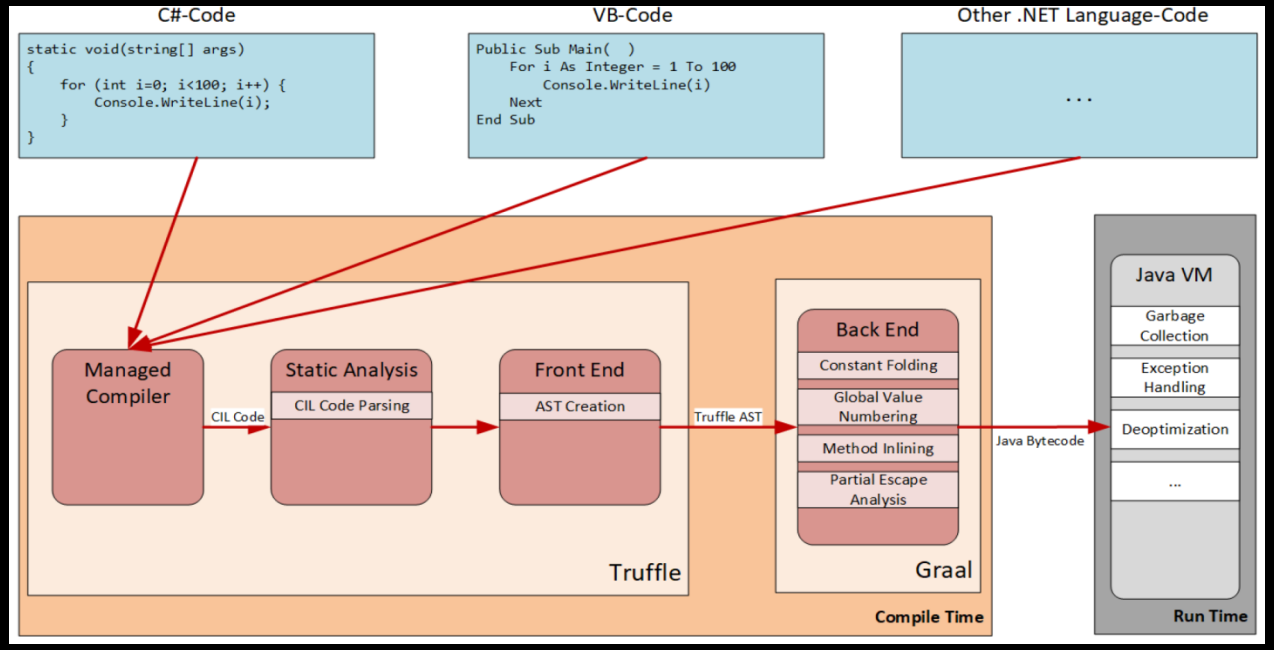
- …

4) Renode on GraalVM

Scheme 1

- Running Renode on GraalVM
- Truffle CIL Interpreter

https://githubmemory.com/repo/jagotu/BACIL
https://epub.jku.at/obvulihs/download/pdf/5473678?originalFilename=true(not open-sourced, and poor performance…)



Scheme 2

- Rewriting Renode with a JVM language…

5) Summary

- Project Renode does not have built-in support for building and running on ARM(includes unable to emulate cores for AArch64) by now, and a fair amount of work should be payed for corresponding porting if we want to do it by ourselves;
- Renode has a special release that compiled to Mono/.Net assembly, which is platform-independent. But we still need to overcome several blocking issues so as to make it work properly on RPi4;
- Leveraging Virtualization technologies could be another feasible way;
- "Renode on GraalVM" is awesome if we could implement it…

VI. Wrap-up

- HW/SW co-designed system is the future.
- Virtual Platform plays a very important role in developing an HW/SW co-designed system.
- RISC-V based implementation should be the best practice for an open HW/SW co-designed system in this decade.
- Renode has great potential for HW/SW co-development, especially for RISC-V based solutions.
- There is still much room for improvement of Renode.
- Please look forward to a new talk "Revisiting Renode as a Swiss Army Knife for RISC-V development " against the upcoming Renode 1.13.

Q & A

2021.NET Conf China

开源共建 | 开放创新 | 开发赋能

THANKS!
感谢观看

鲜卑拓跋枫

扫一扫上面的二维码图案，加我微信

Reference

2021.NET Conf China
开源共建 | 开放创新 | 开发赋能

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/

- https://en.wikipedia.org/wiki/Runtime_(program_lifecycle_phase)
- "Renode - a flexible simulator for CI in complex embedded systems", Michael Gielda, LPC2020.
- …