1st OSEDA Workshop

(China, online)

First exploration of D for HW-SW co-designed system

Feng Li (李枫)
hkli2013@126.com
Aug 28, 2022

Agenda

I. Background
- Tech Stack
- Why is D

II. D-based HDL & HVL
- DHDL
- Vlang

III. D for IR
- GyreD
- idjit
- How about DLLVM

IV. Parallel Computing with D
- Libmir
- GDTk
- DCompute
- Rethinking

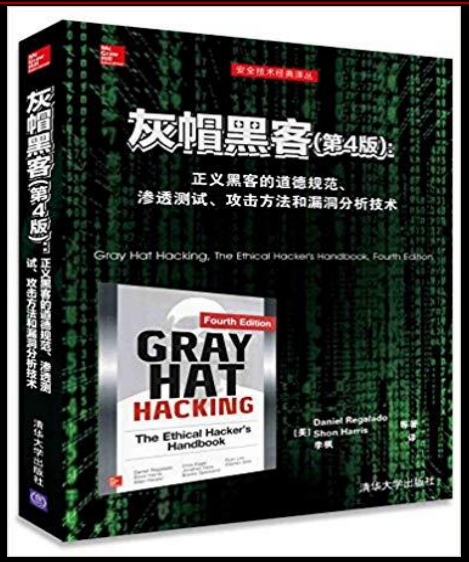
V. D for HW-SW Co-design/FOSS EDA

- D-based DSL
- System Modeling
- D for Virtual Platform

VI. Wrap-up

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(~11 years on Mobile Dev)
- 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…

I. Background

1) Tech Stack

Please may refer to corresponding part in my previous talks as below for related content:

- "Will D be a better system programming language" at OpenInfra Days China 2022(Online).
- "RISC-V based HW-SW System for Edge AI" at RISC-V Summit China 2022(Online).
- Upcoming follow-ups…

HW-SW Co-design

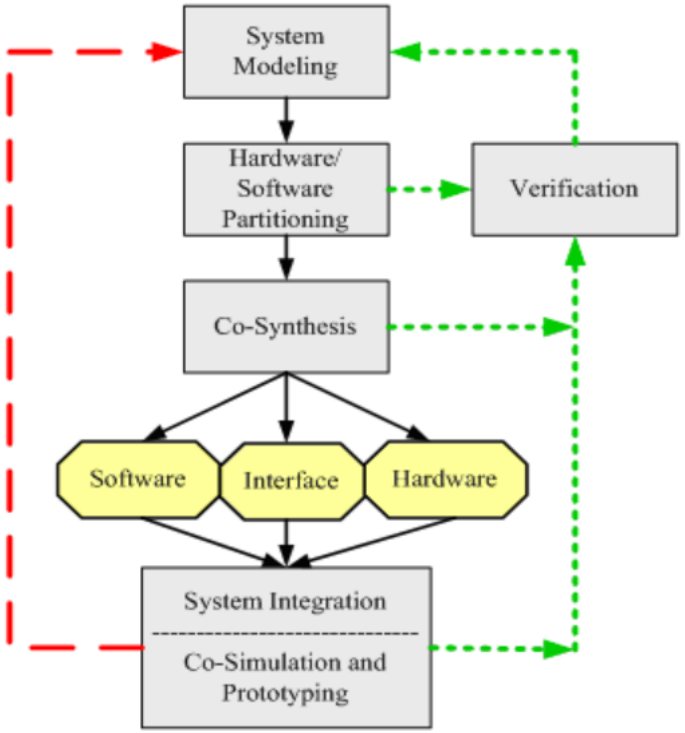
- **https://webhome.cs.uvic.ca/~mserra/HScodesign.html**

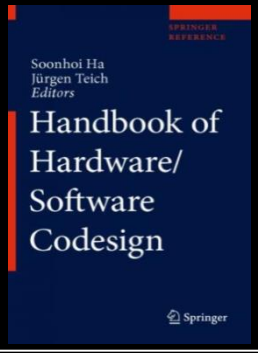
The co-design of HW/SW systems may be viewed as composed of four main phases as illustrated in in the side diagram:

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:

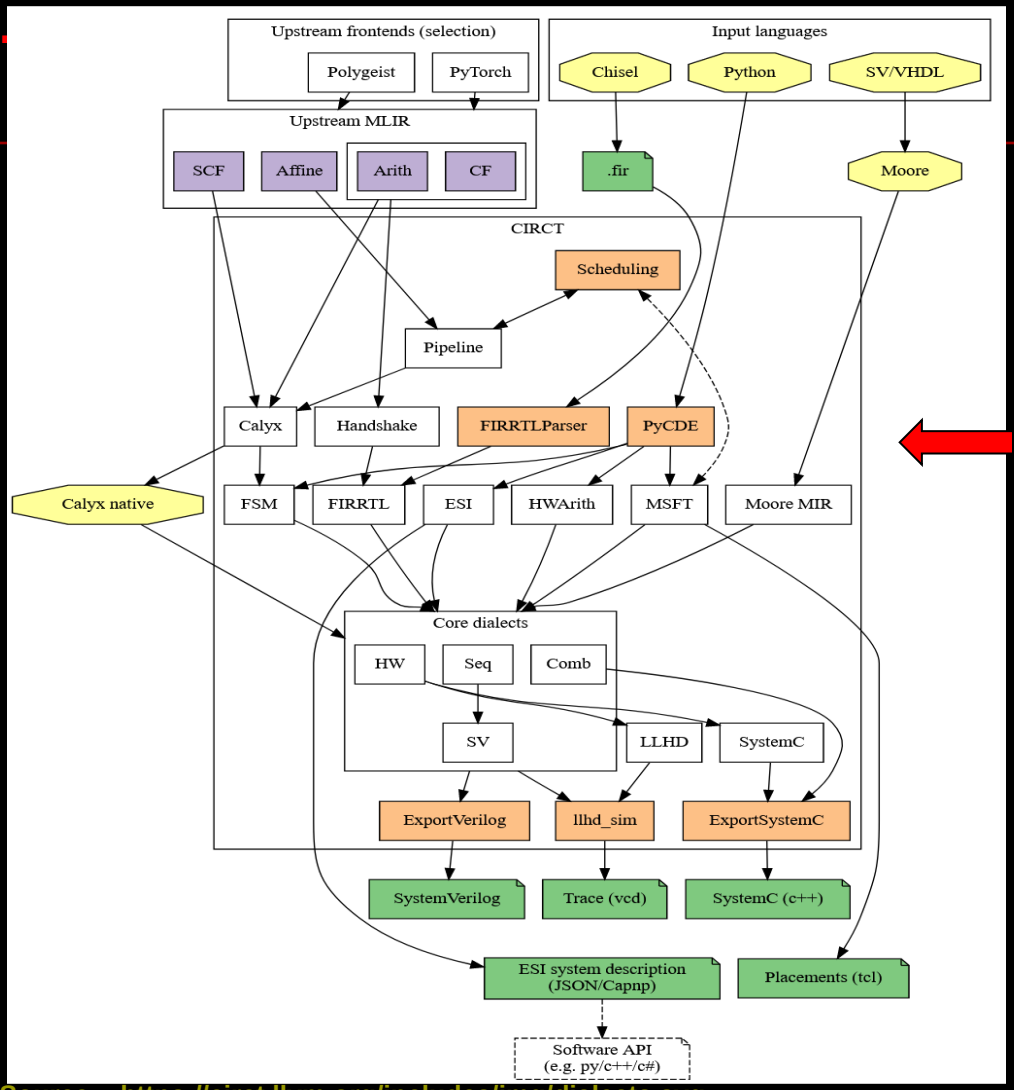
- An existing software implementation of the problem.
- An existing hardware, for example a chip, is present.
- None of the above is given, only specifications leaving an open choice for a model.





CIRCT

- https://circt.llvm.org/ //Circuit IR Compilers and Tools.
- .

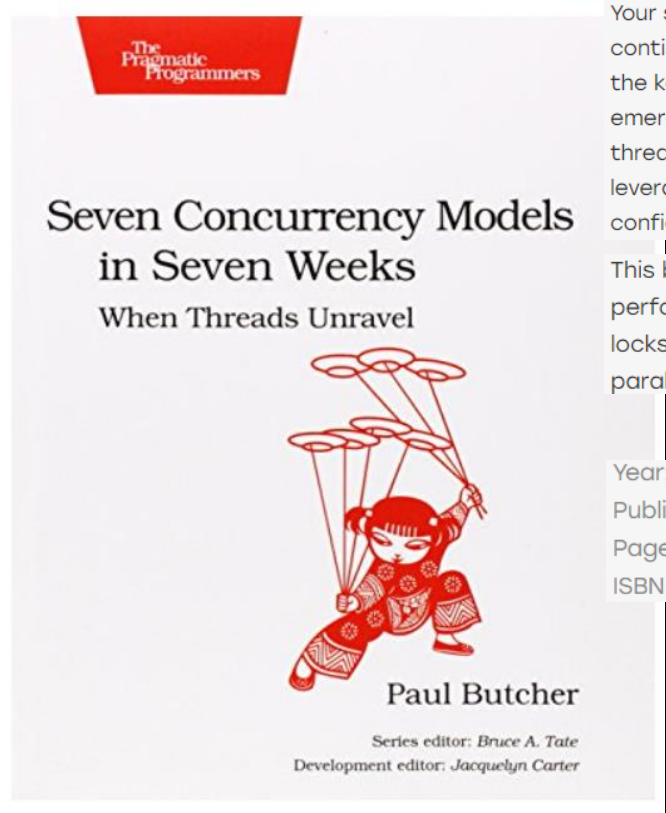


dialects and how they interact

Source: https://circt.llvm.org/includes/img/dialects.svg

Concurrency Models

- Pragmatic Programmers

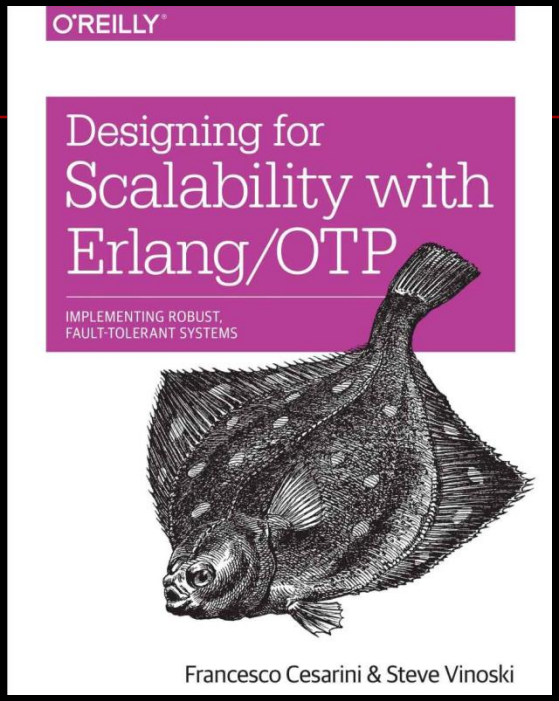


Your software needs to leverage multiple cores, handle thousands of users and terabytes of data, and continue working in the face of both hardware and software failure. Concurrency and parallelism are the keys, and *Seven Concurrency Models in Seven Weeks* equips you for this new world. See how emerging technologies such as actors and functional programming address issues with traditional threads and locks development. Learn how to exploit the parallelism in your computer's GPU and leverage clusters of machines with MapReduce and Stream Processing. And do it all with the confidence that comes from using tools that help you write crystal clear, high-quality code.

This book will show you how to exploit different parallel architectures to improve your code's performance, scalability, and resilience. You'll learn about seven concurrency models: threads and locks, functional programming, separating identity and state, actors, sequential processes, data parallelism, and the lambda architecture.

Year:	2014	Edition:	1
Publisher:	Pragmatic Bookshelf	Language:	english
Pages:	300	ISBN 10:	1937785653
ISBN 13:	9781937785659	Series:	The Pragmatic Programmers

- **Erlang**
  https://en.wikipedia.org/wiki/Erlang_(programming_language)
  Inspired by Actor Model and CSP(Communicating Sequential Processes)



- https://en.wikipedia.org/wiki/Actor_mode
- https://en.wikipedia.org/wiki/Concurrency_(computer_science)
- https://en.wikipedia.org/wiki/Concurrent_computing
- https://en.wikipedia.org/wiki/List_of_concurrent_and_parallel_programming_languages
- ...

UVM

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

The **Universal Verification Methodology** (UVM) is a standardized methodology for verifying integrated circuit designs. UVM is derived mainly from the OVM (Open Verification Methodology) which was, to a large part, based on the eRM (e Reuse Methodology) for the e Verification Language developed by Verisity Design in 2001. The UVM class library brings much automation to the SystemVerilog language such as sequences and data automation features (packing, copy, compare) etc., and unlike the previous methodologies developed independently by the simulator vendors, is an Accellera standard with support from multiple vendors: Aldec, Cadence, Mentor Graphics, Synopsys, Xilinx Simulator(XSIM).

History [edit]

In December 2009, a technical subcommittee of Accellera — a standards organization in the electronic design automation (EDA) industry — voted to establish the UVM and decided to base this new standard on the Open Verification Methodology (OVM-2.1.1),[1] a verification methodology developed jointly in 2007 by Cadence Design Systems and Mentor Graphics.

On February 21, 2011, Accellera approved the 1.0 version of UVM.[2] UVM 1.0 includes a Reference Guide, a Reference Implementation in the form of a SystemVerilog base class library, and a User Guide.[2]

Definitions [edit]

- Agent - A container that emulates and verifies DUT devices
- Blocking - An interface that blocks tasks from other interfaces until it completes
- DUT - Device under test, what you are actually testing
- DUV - Device Under Verification
- Component - A portion of verification intellectual property that has interfaces and functions.
- Transactor - see component
- Verification Environment Configuration - those settings in the DUT and environment that are alterable while the simulation is running
- VIP - verification intellectual property

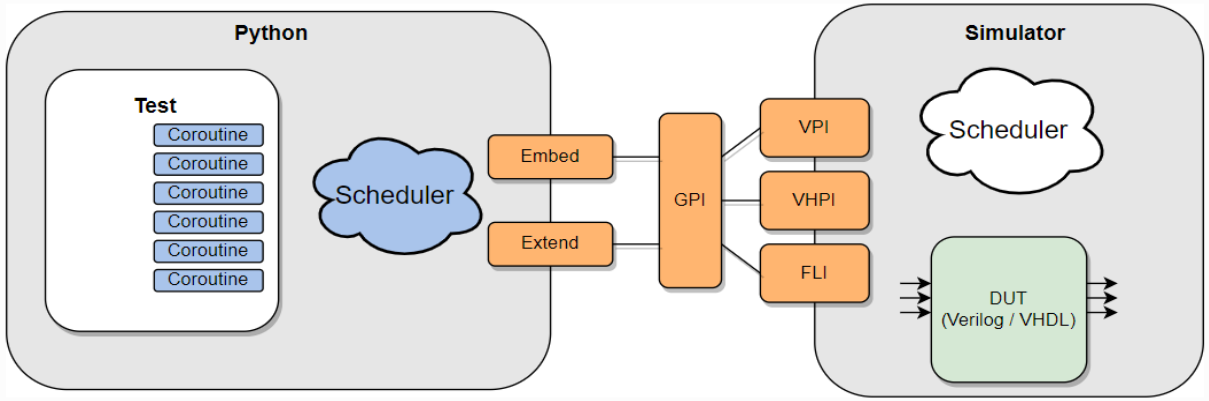
…

- https://www.accellera.org/community/uvm/
- The latest version of UVM is 1.2, and 2.0 is on the way.
  https://www.accellera.org/downloads/standards/uvm
- …

Cocotb

- https://www.cocotb.org/
  A Coroutine based Cosimulation TestBench environment for verifying VHDL and SystemVerilog RTL using Python.
- https://github.com/cocotb/cocotb
- How it works

A typical cocotb testbench requires no additional RTL code. The Design Under Test (DUT) is instantiated as the toplevel in the simulator without any wrapper code. cocotb drives stimulus onto the inputs to the DUT (or further down the hierarchy) and monitors the outputs directly from Python. Note that cocotb can not instantiate HDL blocks - your DUT must be complete.



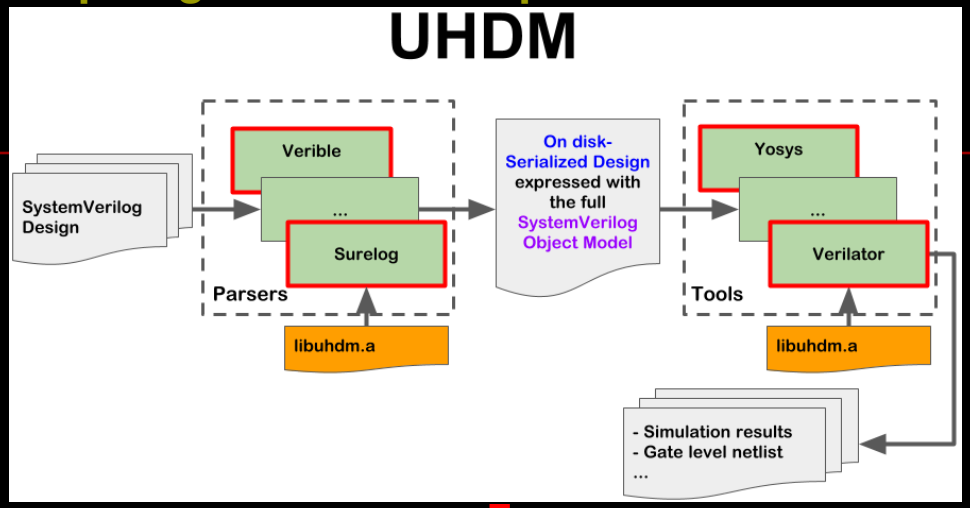
A test is simply a Python function. At any given time either the simulator is advancing time or the Python code is executing. The `await` keyword is used to indicate when to pass control of execution back to the simulator. A test can spawn multiple coroutines, allowing for independent flows of execution.

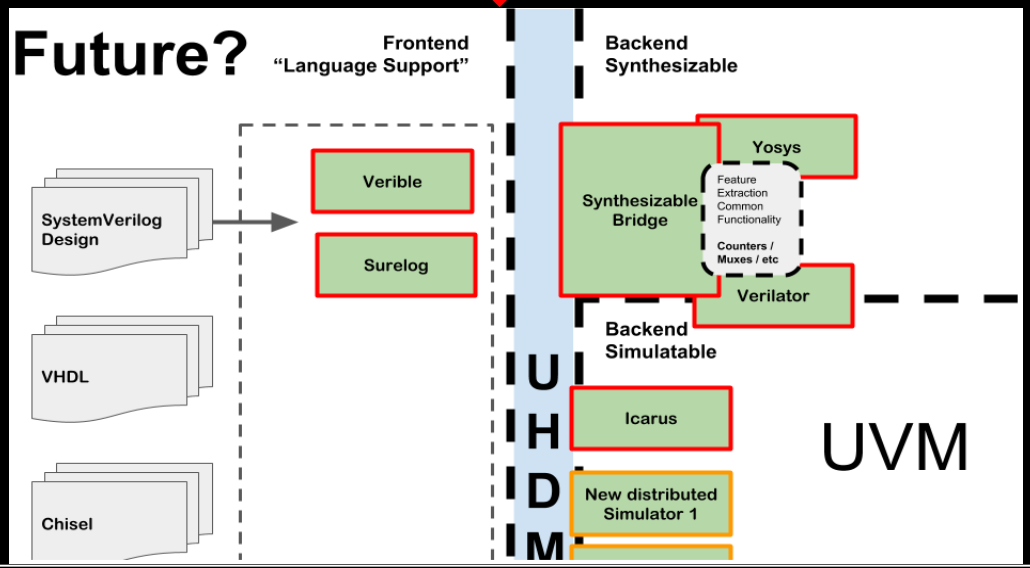
Source: https://docs.cocotb.org/en/stable/

- https://www.reddit.com/r/FPGA/comments/v8fd63/cocotb_in_python_vs_uvm_in_systemverilog/
- ...

UHDM Workflow

- https://github.com/chipsalliance/UHDM





FPGA in the Cloud

- **https://fires.im/**

  Easy-to-use, FPGA-accelerated Cycle-accurate Hardware Simulation in the Cloud

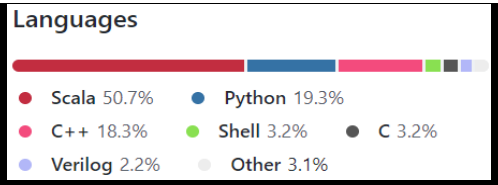
  FireSim is an open-source cycle-accurate FPGA-accelerated full-system hardware simulation platform that runs on cloud FPGAs (Amazon EC2 F1). FireSim is actively developed in the Berkeley Architecture Research Group in the Electrical Engineering and Computer Sciences Department at the University of California, Berkeley.

- **What can I simulate with FireSim?**

  FireSim can simulate arbitrary hardware designs written in Chisel or designs that can be transformed into FIRRTL (including early work on supporting Verilog designs via Yosys's Verilog to FIRRTL flow). With FireSim, you can write your own RTL (processors, accelerators, etc.) and run it at near-FPGA-prototype speeds on cloud FPGAs, while obtaining cycle-accurate performance results (i.e. matching what you would find if you taped-out a chip). Depending on the hardware design and the simulation scale, FireSim simulations run at **10s to 100s of MHz**. You can also integrate custom software models for components that you don't want/need to write as RTL.

  FireSim was originally developed to simulate datacenters by combining open RTL for RISC-V processors with a custom cycle-accurate network simulation. By default, FireSim provides all the RTL and models necessary to **cycle-exactly** simulate from **one to thousands of multi-core compute nodes**, derived directly from **silicon-proven** and **open** target-RTL (RISC-V Rocket Chip and BOOM), with an optional **cycle-accurate network simulation** tying them together. FireSim also provides a Linux distribution that is compatible with the RISC-V systems it simulates and automates the process of including new workloads into this Linux distribution. These simulations run fast enough to interact with Linux on the simulated system at the command line, like a real computer. Users can even SSH into simulated systems in FireSim and access the Internet from within them.

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



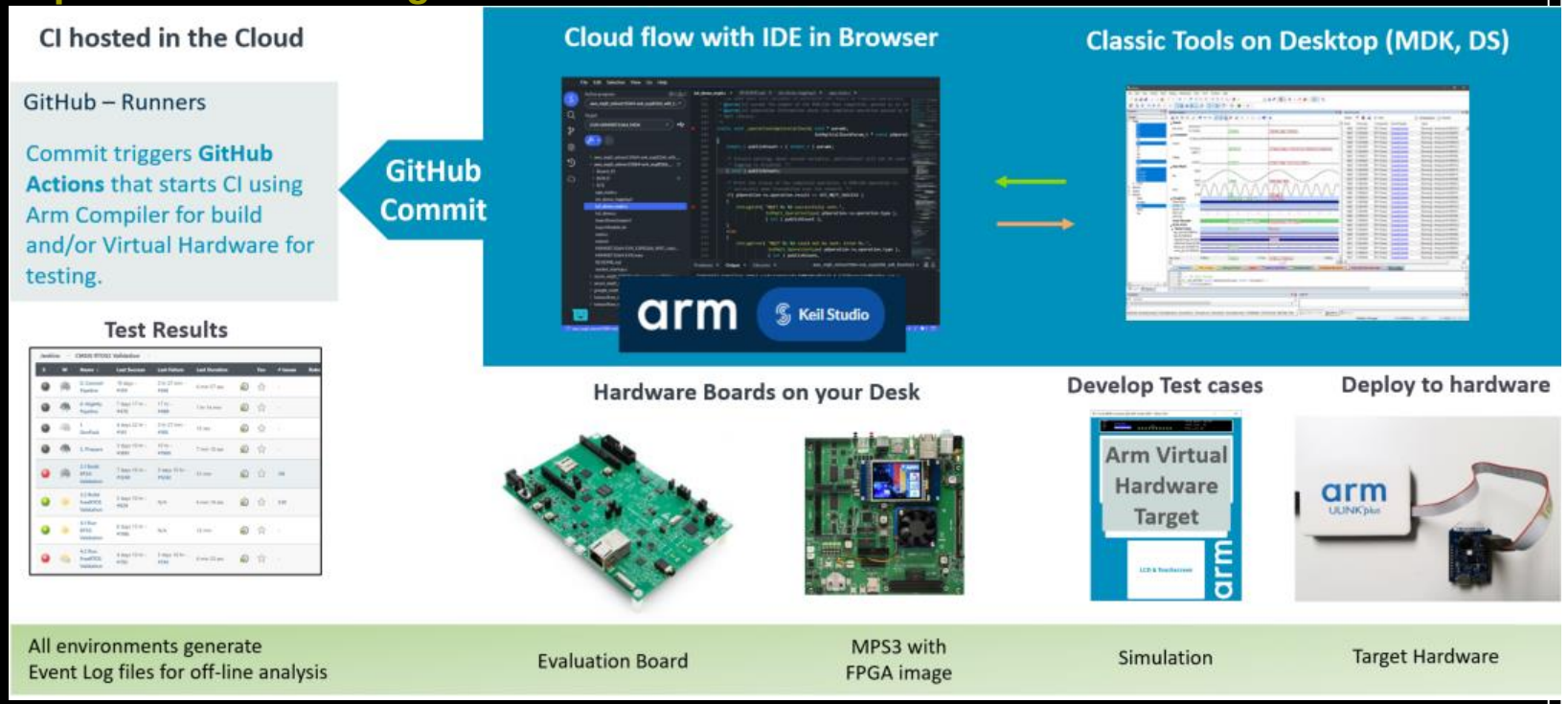
- **https://github.com/chipsalliance/rocket-chip**

AVH

- **https://avh.arm.com/**

Arm Virtual Hardware (AVH) provides a step change in the evolution of Arm's modeling technology. Arm Virtual Hardware delivers test platforms for developers to verify and validate embedded and IoT applications during the complete software design cycle. Multiple modeling technologies are provided that remove the complexity of building and configuring board farms. This enables modern, agile, cloud native software development practices, such as continuous integration and continuous development CI/CD (DevOps) and MLOps workflows.

https://arm-software.github.io/AVH/main/overview/html/index.html



https://github.com/ARM-software/AVH

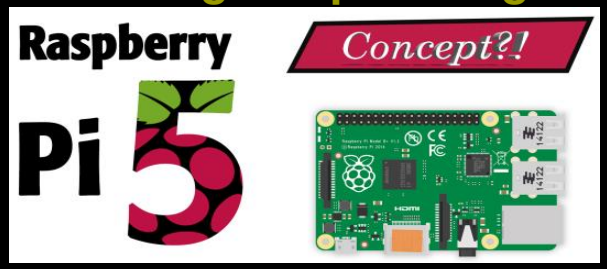
...

Testbed

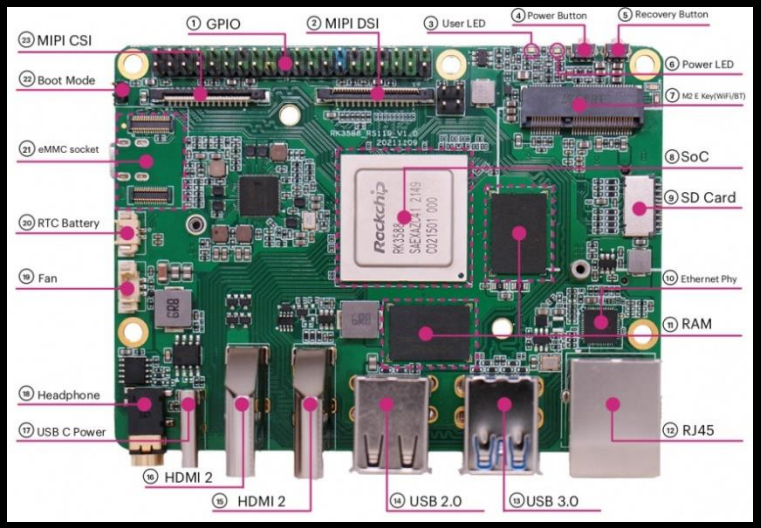
- Raspberry Pi 4(8GB LPDDR4) with Fedora 36

```
[mydev@fedora /]$ uname -a
Linux fedora 5.18.19-200.fc36.aarch64 #1 SMP PREEMPT_DYNAMIC Sun Aug 21 15:31:08 UTC 2022 aarch64 aarch64 aarch64 GNU/Linux
[mydev@fedora /]$
[mydev@fedora /]$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7828         712         619           7        6496        6690
Swap:           7827           4        7823
[mydev@fedora /]$
[mydev@fedora /]$ clang -v
clang version 14.0.5 (Fedora 14.0.5-1.fc36)
Target: aarch64-redhat-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
Found candidate GCC installation: /usr/bin/../lib/gcc/aarch64-redhat-linux/12
Selected GCC installation: /usr/bin/../lib/gcc/aarch64-redhat-linux/12
Candidate multilib: .;@m64
Selected multilib: .;@m64
[mydev@fedora /]$
[mydev@fedora /]$ ldc2 --version
LDC - the LLVM D compiler (1.30.0):
  based on DMD v2.100.1 and LLVM 14.0.3
  built with LDC - the LLVM D compiler (1.30.0)
  Default target: aarch64-unknown-linux-gnu
  Host CPU: cortex-a72
  http://dlang.org - http://wiki.dlang.org/LDC
```

- Switching to upcoming development boards



&



RK3588-based dev boards and more…

2) Why is D

I am here

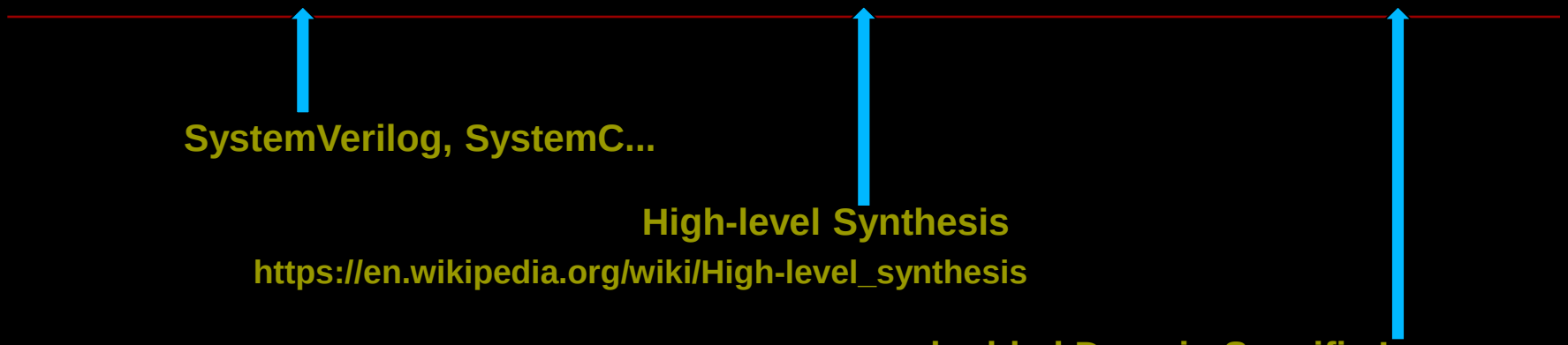
- One idea: Why Modern Alternative Languages Never Replace C/C++

  https://levelup.gitconnected.com/why-modern-alternative-languages-never- replace-c-c-cbf0afc5f1dc

- But…

Evolution of HDLs

- https://en.wikipedia.org/wiki/Hardware_description_language
- https://hdl.github.io/awesome/items/
- Verilog/VHDL → HLS → eDSL



Typical eDSLs

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

A desired system language for HW-SW co-development

- Comparable performance to C/C++/Rust/Go.
- Guarantee memory-safety and thread-safety as Rust.
- Low learning curve and high productivity, especially when compared to C++/Rust, while close to the level of Python/Java is mostly preferred.
- Support Multi-paradigm programming, especially for Functional and Metaprogramming, as well as Concurrent.
- Good interoperability for most of the popular programming languages.
- Built-in support for concurrency that closer to Go.
- A self-hosting compilers is mostly preferred.
- The ability of bare-metal programming.
- Can be used for Linux Kernel development, and more.
- Natively support for Heterogeneous Parallel Computing.
- Easier to implement new DSLs base on it.
- With high-level abstraction ability for HW/SW Co-design, Co-development, Co-simulation, and Co-debugging etc.
- Come with a certain foundation of ecosystem.

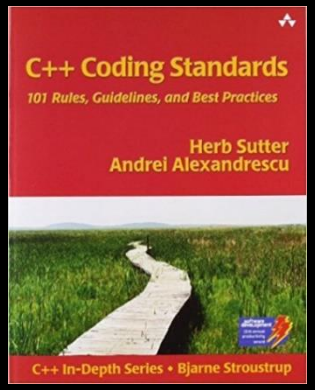
Through a comprehensive evaluation, we do think  has tremendous potential to meet all of our needs to a great extent.

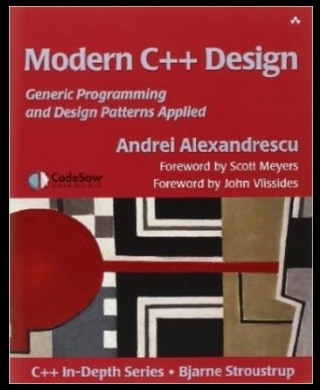
Designed by Experts

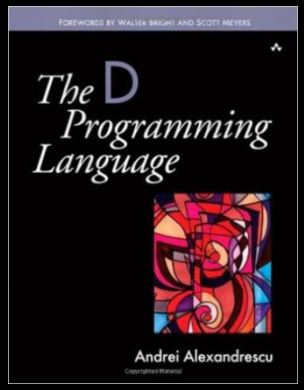
- https://en.wikipedia.org/wiki/D_(programming_language)
- https://en.wikipedia.org/wiki/Walter_Bright
  http://digitalmars.com/

Digital Mars D compiler
Digital Mars C compiler
Digital Mars C++ compiler

- https://en.wikipedia.org/wiki/Andrei_Alexandrescu
  http://erdani.org/







- **Origins of the D Programming Language**
  https://dl.acm.org/doi/abs/10.1145/3386323

Major Design Goals

Everything in designing a language is a tradeoff. Keeping some principles in mind will help to make the right decisions.

1. Enable writing fast, effective code in a straightforward manner.
2. Make it easier to write code that is portable from compiler to compiler, machine to machine, and operating system to operating system. Eliminate undefined and implementation defined behaviors as much as practical.
3. Provide syntactic and semantic constructs that eliminate or at least reduce common mistakes. Reduce or even eliminate the need for third party static code checkers.
4. Support memory safe programming.
5. Support multi-paradigm programming, i.e. at a minimum support imperative, structured, object oriented, generic and even functional programming paradigms.
6. Make doing things the right way easier than the wrong way.
7. Have a short learning curve for programmers comfortable with programming in C, C++ or Java.
8. Provide low level bare metal access as required. Provide a means for the advanced programmer to escape checking as necessary.
9. Be compatible with the local C application binary interface.
10. Where D code looks the same as C code, have it either behave the same or issue an error.
11. Have a context-free grammar. Successful parsing must not require semantic analysis.
12. Easily support writing internationalized applications - Unicode everywhere.
13. Incorporate Contract Programming and unit testing methodology.
14. Be able to build lightweight, standalone programs.
15. Reduce the costs of creating documentation.
16. Provide sufficient semantics to enable advances in compiler optimization technology.
17. Cater to the needs of numerical analysis programmers.
18. Obviously, sometimes these goals will conflict. Resolution will be in favor of usability.

Source: https://dlang.org/overview.html

Vision

- https://github.com/dlang/vision-document
- https://github.com/dlang-community/awesome-d
- http://wiki.dlang.org/
- http://code.dlang.org/
- https://wiki.dlang.org/Open_Source_Projects
- ...

Pros & Cons

- Pros

Development Mode	mainly community-driven https://github.com/dlang/DIPs/ https://dlang.org/foundation/
Productivity	a combination of C++/C/Java/Scala/Python…
Flexibility	e.g., hybird(auto/manually) memory management
Binary-compatible with C	essentially, this means that it should be possible to compile a C source file with a C compiler and combine the output into a program that is written in D and compiled with a D compiler, or vice versa
Low level programming	pointer, inline assembler…
Interop	easily interface with legacy code written in C/C++/Lua/Python…
Multi-paradigm	includes but not limited to imperative, functional, object-oriented, metaprogramming, and concurrent
Good support for HPC	e.g., DCompute in LDC, project Mir…
Built-in unit test support	QA friendly
…	…

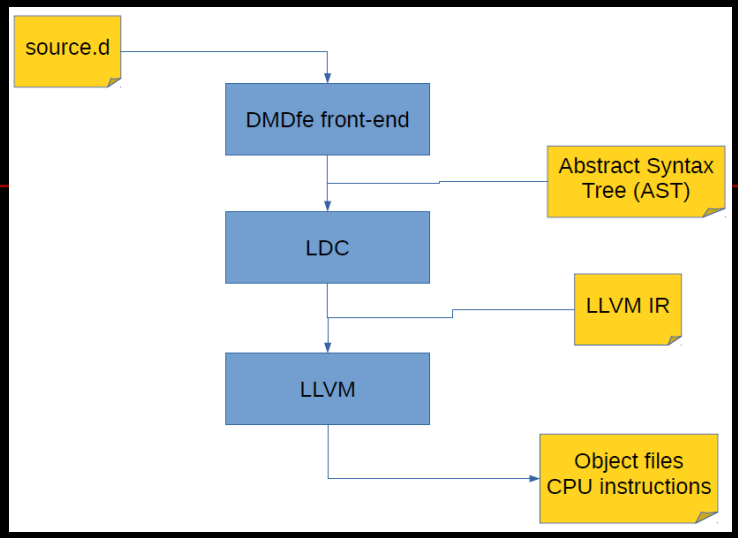
- *Cons*

  Lack of popular open source projects;
  Lack of good application frameworks/libraries;
  Not as mature as commercial products, e.g. GC algorithms etc;
  It seems that some of the D-based open-source projects may not be well-tested outside X86;
  The ecosystem is still weak when compared with that of C++, Go, Rust…

should pay more attention on ARM & RISC-V.

LDC

```
[mydev@fedora ldc-master]$ tree -L 1 .
.
├── bitrise.yml
├── CHANGELOG.md
├── cmake
├── CMakeCPack.cmake
├── CMakeLists.txt
├── dmd
├── docs
├── Doxyfile
├── driver
├── gen
├── ir
├── ldc2.conf.in
├── ldc2_install.conf.in
├── LDC2-master.geany
├── ldc2_phobos.conf.in
├── LICENSE
├── packaging
├── README.md
├── runtime
├── tests
├── tools
└── utils
```



Source: "LLVM-backed goodies in LDC", Johan Engelen, DConf 2018.

- https://github.com/ldc-developers/ldc/releases

ldc-1.30.0-src.zip	10.3 MB	21 Jul 2022
ldc2-1.30.0-android-aarch64.tar.xz	30.1 MB	21 Jul 2022
ldc2-1.30.0-android-armv7a.tar.xz	27.6 MB	21 Jul 2022
ldc2-1.30.0-freebsd-x86_64.tar.xz	17.3 MB	21 Jul 2022
ldc2-1.30.0-linux-aarch64.tar.xz	51.3 MB	21 Jul 2022
ldc2-1.30.0-linux-x86_64.tar.xz	62.3 MB	21 Jul 2022
ldc2-1.30.0-osx-arm64.tar.xz	65.5 MB	21 Jul 2022
ldc2-1.30.0-osx-universal.tar.xz	136 MB	21 Jul 2022
ldc2-1.30.0-osx-x86_64.tar.xz	72.8 MB	21 Jul 2022
ldc2-1.30.0-windows-multilib.7z	50.1 MB	21 Jul 2022
ldc2-1.30.0-windows-multilib.exe	53.1 MB	21 Jul 2022
ldc2-1.30.0-windows-x64.7z	36 MB	21 Jul 2022
ldc2-1.30.0-windows-x86.7z	35.1 MB	21 Jul 2022
ldc2-1.30.0.sha256sums.txt	1.31 KB	21 Jul 2022

build LDC from src on RPi4

https://wiki.dlang.org/Building_LDC_from_source

Run the following commands to configure and build LDC and its default libraries:

```
# Generate Ninja build files
mkdir build-ldc && cd build-ldc # using a fresh new build dir is highly recommended whenever re-invoking CMake
cmake -G Ninja ../ldc \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=$PWD/../install-ldc \
  -DLLVM_ROOT_DIR=$PWD/../install-llvm \                    # not needed if using a distro LLVM package
  -DD_COMPILER=$PWD/../ldc2-1.17.0-linux-x86_64/bin/ldmd2 # not needed if host D compiler (ldmd2/dmd/gdmd) is found in PATH or if building 0.17 ltsmaster

# Build; use -j<N> to limit parallelism if running out of memory. The binaries end up in bin/.
ninja
# Optional: install LDC to the CMAKE_INSTALL_PREFIX directory above
ninja install
```

output:

```
[mydev@fedora ldc-master]$ tree -L 1 build/Install/
build/Install/
├── bin
├── etc
├── include
└── lib
```

```
[mydev@fedora ldc-master]$ tree -L 1 build/Install/bin
build/Install/bin
├── ldc2
├── ldc-build-runtime
├── ldc-profdata
├── ldc-prune-cache
└── ldmd2
```

```
[mydev@fedora ldc-master]$ tree -L 1 build/Install/lib
build/Install/lib
├── ldc_rt.dso.o
├── libdruntime-ldc.a
├── libdruntime-ldc-debug.a
├── libdruntime-ldc-debug-shared.so → libdruntime-ldc-debug-shared.so.100
├── libdruntime-ldc-debug-shared.so.100 → libdruntime-ldc-debug-shared.so.100.1
├── libdruntime-ldc-debug-shared.so.100.1
├── libdruntime-ldc-shared.so → libdruntime-ldc-shared.so.100
├── libdruntime-ldc-shared.so.100 → libdruntime-ldc-shared.so.100.1
├── libdruntime-ldc-shared.so.100.1
├── libphobos2-ldc.a
├── libphobos2-ldc-debug.a
├── libphobos2-ldc-debug-shared.so → libphobos2-ldc-debug-shared.so.100
├── libphobos2-ldc-debug-shared.so.100 → libphobos2-ldc-debug-shared.so.100.1
├── libphobos2-ldc-debug-shared.so.100.1
├── libphobos2-ldc-shared.so → libphobos2-ldc-shared.so.100
├── libphobos2-ldc-shared.so.100 → libphobos2-ldc-shared.so.100.1
└── libphobos2-ldc-shared.so.100.1
```

2) D-based HDL & HVL

2.1 DHDL

- https://github.com/luismarques/dhdl

  The D Hardware Design Language.

- Model

  - Built-in data types and their semantics
  - Expressing combinational circuits using D operators
  - Maintaining state with registers
  - The semantics of assignment and concurrent execution
  - Grouping logic into hardware blocks

  Source: https://dconf.org/2017/talks/marques.html

- Src

Stats (master branch and last commit: 3d4514fda62dd65ff591f3a8ca385865561939b8)

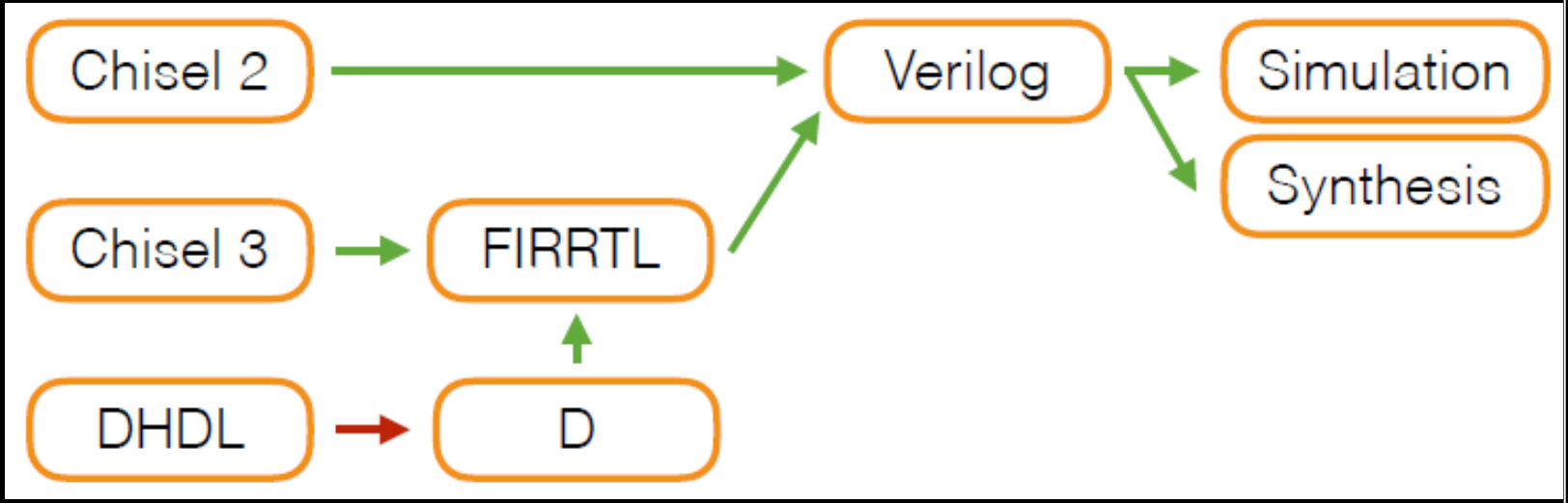
```
[mydev@fedora dhdl]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 D                      25         4416         3434          157          825
 JSON                    2           19           19            0            0
 Markdown                1           34            0           22           12
===============================================================================
 Total                  28         4469         3453          179          837
===============================================================================
```

```
mydev@fedora dhdl]$ tree .
.
├── dub.json
├── dub.selections.json
├── README.md
├── source
│   └── dhdl
│       ├── dhdl.d
│       ├── firrtl.d
│       ├── lib
│       │   ├── coupling.d
│       │   └── queuing.d
│       ├── package.d
│       ├── sim.d
│       ├── testing.d
│       ├── tests
│       │   ├── arithmetic.d
│       │   ├── bundles.d
│       │   ├── clocking.d
│       │   ├── composition.d
│       │   ├── concatenation.d
│       │   ├── conditionals.d
│       │   ├── coupling.d
│       │   ├── helloworld.d
│       │   ├── indexing.d
│       │   ├── matching.d
│       │   ├── memories.d
│       │   ├── operators.d
│       │   ├── registers.d
│       │   ├── slicing.d
│       │   ├── vectors.d
│       │   └── wires.d
│       ├── util.d
│       └── verilator.d
```

Design

Creating Hardware

- Approach 1: High-level synthesis
  Approach 2: Event-based
  Approach 3: RTL builder
- VHDL & Verilog: Follow mostly approach 2
- DHDL (and Chisel, SpinalHDL, etc.): Follows mostly approach 3



Source: "DHDL: The D Hardware Description Language", Luís Marques, DConf 2017.

Hello World

helloworld.dhdl

```
circuit HelloWorld
{
    port in bool a;
    port out bool b;

    b := a;
}
```


```
unittest
{
    auto hello = peekPokeTester!HelloWorld;

    hello.a = false;
    hello.eval();
    assert(hello.b == false);

    hello.a = true;
    hello.eval();
    assert(hello.b == true);
}
```

HelloWorld.fir

```
circuit HelloWorld :
    module HelloWorld :
        input reset : UInt<1>
        input clock : Clock
        input a : UInt<1>
        output b : UInt<1>

        b is invalid

        b <= a
```

HelloWorld.v

```
module HelloWorld(
  input   reset,
  input   clock,
  input   a,
  output  b
);
  assign b = a;
endmodule
```

Source: "DHDL: The D Hardware Description Language", Luís Marques, DConf 2017.

Current status on RPi4

```
[mydev@fedora dhdl]$ export PATH=$PATH:/opt/MyWorkSpace/MyProjs/HW-EDA/RISC-V/Chisel/firrtl/utils/bin
[mydev@fedora dhdl]$ dub test
Fetching openmethods 1.1.0 (getting selected version)...
Generating test runner configuration 'dhdl-test-library' for 'library' (library).
Performing "unittest" build using /opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 for aarch64, arm_hardfloat.
openmethods 1.1.0: building configuration "library"...
/home/mydev/.dub/packages/openmethods-1.1.0/openmethods/source/openmethods.d(938,7): Deprecation: foreach: loop index implicit
ly converted from `size_t` to `int`
/home/mydev/.dub/packages/openmethods-1.1.0/openmethods/source/openmethods.d(1348,7): Deprecation: foreach: loop index implici
tly converted from `size_t` to `int`
/home/mydev/.dub/packages/openmethods-1.1.0/openmethods/source/openmethods.d(1362,13): Deprecation: foreach: loop index implic
itly converted from `size_t` to `int`
/home/mydev/.dub/packages/openmethods-1.1.0/openmethods/source/openmethods.d(1393,7): Deprecation: foreach: loop index implici
tly converted from `size_t` to `int`
/home/mydev/.dub/packages/openmethods-1.1.0/openmethods/source/openmethods.d(1414,7): Deprecation: foreach: loop index implici
tly converted from `size_t` to `int`
/home/mydev/.dub/packages/openmethods-1.1.0/openmethods/source/openmethods.d(1566,9): Deprecation: foreach: loop index implici
tly converted from `size_t` to `int`
dhdl ~master: building configuration "dhdl-test-library"...
source/dhdl/firrtl.d(524,53): Error: cannot cast expression `literal` of type `Nullable!ulong` to `long`
source/dhdl/testing.d(72,30): Error: constructor `dhdl.verilator.VPort!(1, false).VPort.this(void* portAddr)` is not callable
using argument types `(Nullable!(VPort!(1, false)))`
source/dhdl/testing.d(72,30):        cannot pass argument `this._defaultClock` of type `Nullable!(VPort!(1, false))` to parame
ter `void* portAddr`
source/dhdl/testing.d(19,18): Error: template instance `dhdl.testing.PeekPokeTester!(Queue!(UInt!16, 8))` error instantiating
source/dhdl/lib/queuing.d(120,34):        instantiated from here: `peekPokeTester!(Queue!(UInt!16, 8), true, bool, bool)`
source/dhdl/testing.d(72,30): Error: constructor `dhdl.verilator.VPort!(1, false).VPort.this(void* portAddr)` is not callable
using argument types `(Nullable!(VPort!(1, false)))`
source/dhdl/testing.d(72,30):        cannot pass argument `this._defaultClock` of type `Nullable!(VPort!(1, false))` to parame
ter `void* portAddr`
source/dhdl/testing.d(19,18): Error: template instance `dhdl.testing.PeekPokeTester!(Arithmetic)` error instantiating
source/dhdl/tests/arithmetic.d(34,14):        instantiated from here: `peekPokeTester!(Arithmetic, true)`
source/dhdl/testing.d(72,30): Error: constructor `dhdl.verilator.VPort!(1, false).VPort.this(void* portAddr)` is not callable
using argument types `(Nullable!(VPort!(1, false)))`
...
source/dhdl/testing.d(19,18): Error: template instance `dhdl.testing.PeekPokeTester!(Outer)` error instantiating
source/dhdl/tests/composition.d(47,14):        instantiated from here: `peekPokeTester!(Outer, true)`
source/dhdl/testing.d(72,30): Error: constructor `dhdl.verilator.VPort!(1, false).VPort.this(void* portAddr)` is not callable
using argument types `(Nullable!(VPort!(1, false)))`
source/dhdl/testing.d(72,30):        cannot pass argument `this._defaultClock` of type `Nullable!(VPort!(1, false))` to parame
ter `void* portAddr`
source/dhdl/testing.d(19,18): Error: template instance `dhdl.testing.PeekPokeTester!(Concatenation)` error instantiating
/opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 failed with exit code 1.
[mydev@fedora dhdl]$
```

Tried to upgrade the version of “openmethods”:

openmethods: https://github.com/jll63/openmethods.d

```
[mydev@fedora dhdl]$ git diff
diff --git a/dub.json b/dub.json
index 44ce515..dfab60a 100644
--- a/dub.json
+++ b/dub.json
@@ -8,6 +8,6 @@
     "license": "BSL-1.0",
     "versions": ["HWTests"],
     "dependencies": {
-        "openmethods": "~>1.1.0"
+        "openmethods": "~>1.3.3"
     }
-}
\ No newline at end of file
+}
[mydev@fedora dhdl]$
```

```
[mydev@fedora dhdl]$ dub upgrade
Upgrading project in /opt/MyWorkSpace/MyProjs/HW-EDA/Languages/D/HDL/DHDL/Official/dhdl
Fetching openmethods 1.3.3 (getting selected version)...
[mydev@fedora dhdl]$
[mydev@fedora dhdl]$ git diff
diff --git a/dub.json b/dub.json
index 44ce515..dfab60a 100644
--- a/dub.json
+++ b/dub.json
@@ -8,6 +8,6 @@
     "license": "BSL-1.0",
     "versions": ["HWTests"],
     "dependencies": {
-        "openmethods": "~>1.1.0"
+        "openmethods": "~>1.3.3"
     }
-}
\ No newline at end of file
+}
diff --git a/dub.selections.json b/dub.selections.json
index 406e979..2375370 100644
--- a/dub.selections.json
+++ b/dub.selections.json
@@ -1,6 +1,7 @@
 {
         "fileVersion": 1,
         "versions": {
-                "openmethods": "1.1.0"
+                "bolts": "1.8.0",
+                "openmethods": "1.3.3"
         }
 }
[mydev@fedora dhdl]$
```

```
[mydev@fedora dhdl]$ dub test
Generating test runner configuration 'dhdl-test-library' for 'library' (library).
Performing "unittest" build using /opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 for aarch64, arm_hardfloat.
bolts 1.8.0: target for configuration "unittest" is up to date.
openmethods 1.3.3: building configuration "unittest"...
dhdl ~master: building configuration "dhdl-test-library"...
mixins.txt(341,84): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(367,84): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(385,102): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(394,102): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(404,30): Error: undefined identifier `firrtl`
mixins.txt(405,5): Error: alias `dhdl.firrtl.emit` conflicts with alias `dhdl.firrtl.emit` at mixins.txt(404,5)
mixins.txt(405,30): Error: undefined identifier `firrtl`
mixins.txt(408,5): Error: alias `dhdl.firrtl.emit` conflicts with alias `dhdl.firrtl.emit` at mixins.txt(405,5)
mixins.txt(408,30): Error: undefined identifier `firrtl`
mixins.txt(409,5): Error: alias `dhdl.firrtl.emit` conflicts with alias `dhdl.firrtl.emit` at mixins.txt(408,5)
mixins.txt(409,30): Error: undefined identifier `firrtl`
mixins.txt(412,36): Error: undefined identifier `firrtl`
mixins.txt(413,5): Error: alias `dhdl.firrtl.emitSymbol` conflicts with alias `dhdl.firrtl.emitSymbol` at mixins.txt(412,5)
mixins.txt(413,36): Error: undefined identifier `firrtl`
mixins.txt(416,37): Error: undefined identifier `firrtl`
mixins.txt(417,5): Error: alias `dhdl.firrtl.composeName` conflicts with alias `dhdl.firrtl.composeName` at mixins.txt(416,5)
mixins.txt(417,37): Error: undefined identifier `firrtl`
/opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 failed with exit code 1.
[mydev@fedora dhdl]$
```

bolts: https://github.com/jll63/openmethods.d

Tried to also upgrade the version of “bolts” via modify that in $SRC_DHDL/dub.selections.json and hack ~/.dub/packages/openmethods-1.3.3:

```
diff --git a/dub.selections.json b/dub.selections.json
index 406e979..beae654 100644
--- a/dub.selections.json
+++ b/dub.selections.json
@@ -1,6 +1,7 @@
 {
        "fileVersion": 1,
        "versions": {
-               "openmethods": "1.1.0"
+               "bolts": "1.8.1",
+               "openmethods": "1.3.3"
        }
 }
```

```
[mydev@fedora dhdl]$ dub upgrade
Upgrading project in /opt/MyWorkSpace/MyProjs/HW-EDA/Languages/D/HDL/DHDL/Official/dhdl
Fetching bolts 1.8.1 (getting selected version)...
[mydev@fedora dhdl]$
```

```
[mydev@fedora dhdl]$ dub test
Generating test runner configuration 'dhdl-test-library' for 'library' (library).
Performing "unittest" build using /opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 for aarch64, arm_hardfloat.
bolts 1.8.1: building configuration "unittest"...
/home/mydev/.dub/packages/bolts-1.8.1/bolts/source/bolts/experimental/refraction.d(28,20): Deprecation: module std.traits is n
ot accessible here, perhaps add 'static import std.traits;'
/home/mydev/.dub/packages/bolts-1.8.1/bolts/source/bolts/experimental/refraction.d(28,1): Deprecation: module std.traits is no
t accessible here, perhaps add 'static import std.traits;'
/home/mydev/.dub/packages/bolts-1.8.1/bolts/source/bolts/experimental/refraction.d(29,20): Deprecation: module std.traits is n
ot accessible here, perhaps add 'static import std.traits;'
/home/mydev/.dub/packages/bolts-1.8.1/bolts/source/bolts/experimental/refraction.d(29,1): Deprecation: module std.traits is no
t accessible here, perhaps add 'static import std.traits;'
openmethods 1.3.3: building configuration "unittest"...
dhdl ~master: building configuration "dhdl-test-library"...
/home/mydev/.dub/packages/bolts-1.8.1/bolts/source/bolts/experimental/refraction.d(29,20): Deprecation: module std.traits is n
ot accessible here, perhaps add 'static import std.traits;'
/home/mydev/.dub/packages/bolts-1.8.1/bolts/source/bolts/experimental/refraction.d(29,1): Deprecation: module std.traits is no
t accessible here, perhaps add 'static import std.traits;'
mixins.txt(389,84): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(416,84): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(434,102): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(443,102): Deprecation: storage class `static` has no effect in type aliases
mixins.txt(453,30): Error: undefined identifier `firrtl`
mixins.txt(454,5): Error: alias `dhdl.firrtl.emit` conflicts with alias `dhdl.firrtl.emit` at mixins.txt(453,5)
mixins.txt(454,30): Error: undefined identifier `firrtl`
mixins.txt(457,5): Error: alias `dhdl.firrtl.emit` conflicts with alias `dhdl.firrtl.emit` at mixins.txt(454,5)
mixins.txt(457,30): Error: undefined identifier `firrtl`
mixins.txt(458,5): Error: alias `dhdl.firrtl.emit` conflicts with alias `dhdl.firrtl.emit` at mixins.txt(457,5)
mixins.txt(458,30): Error: undefined identifier `firrtl`
mixins.txt(461,36): Error: undefined identifier `firrtl`
mixins.txt(462,5): Error: alias `dhdl.firrtl.emitSymbol` conflicts with alias `dhdl.firrtl.emitSymbol` at mixins.txt(461,5)
mixins.txt(462,36): Error: undefined identifier `firrtl`
mixins.txt(465,37): Error: undefined identifier `firrtl`
mixins.txt(466,5): Error: alias `dhdl.firrtl.composeName` conflicts with alias `dhdl.firrtl.composeName` at mixins.txt(465,5)
mixins.txt(466,37): Error: undefined identifier `firrtl`
/opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 failed with exit code 1.
[mydev@fedora dhdl]$
```

Keep working on it…

2.2 Vlang
2.2.1 Overview

- Next Generation Verification Language

Vlang Features at a Glance

Multicore Vlang enables concurrent programming. End user can fine-tune the number of concurrently running threads at module level. Vlang also enables concurrency at a higher abstraction by allowing multiple simulators running in parallel.

Constrained Randomization Full blown and efficient. Concurrency enabled.

UVM Compliance Word-to-word translation of SystemVerilog UVM. More efficient and user-friendly due to generic programming.

Object Oriented Programming Support for function/operator overloading.

Safety and Productivity Automatic Garbage Collection. Exception Handling. Unittests.

Systems Programming Allows low level access to hardware resources. Allows embedded assembly language.

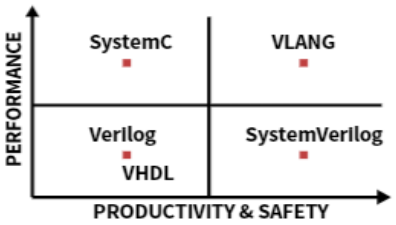
Interface with other Languages Full blown C++ interface. VHPI/VPI bindings with VHDL and SystemVerilog.

Licensing Provided free under open source boost license. Vlang UVM library is available under Apache2 license.

Verification with Vlang

Even as the chip complexity keeps increasing, we continue to rely on same old RTL methodology to design our chips. As a result the abstraction gap between the design and the specification is increasing exponentially.

Vlang attempts to bridge this gap by providing you a high productivity and high performance verification environment.



Higher Productivity means that you take less time in building your verification infrastructure and higher performance means that your regression runs much faster.

If you have ESL as part of your SoC development flow, there are additional reasons that you should use Vlang to verify your ESL models. Vlang supports much better integration with C++ compared to SystemVerilog. Vlang is ABI compatible with C/C++. Vlang also allows you to call any method (including virtual methods) on C++ objects right from Vlang without any boilerplate code. In comparison SystemVerilog DPI interface is limited to C language. As a result any interface between SystemC and SystemVerilog tends to be highly inefficient.

Yet another advantage is that Vlang is free and open source just like your SystemC simulator.

Source: https://fdocuments.in/document/vlang-flyer.html?page=2 (Date Post 28-Dec-2015)

Initial Design

- **Top Down Verification Stack**

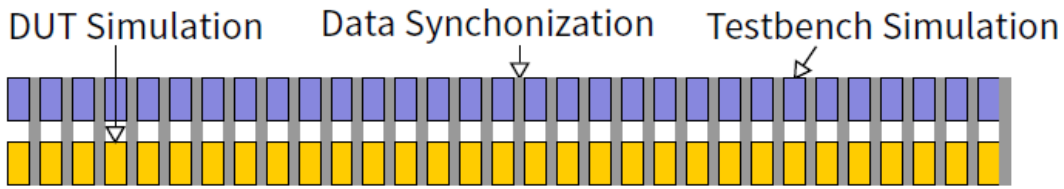
Abstract—IP/SoC verification is a fundamental problem in design cycle which needs support from a suitable and powerful language which must include the latest findings in computer science. State-of-the-art verification languages are decade old, closed source, not software domain, single paradigm, type unsafe, advocate code boilerplate, single core and not supportive to generic programming. This forces verification engineer to use decade old language concepts, using a HW domain language to build a software called test-bench using a single incomplete programming paradigm called object orientation. Furthermore, incomplete support for object orientation added with no support for generative programming forces the user to write redundancy which could be avoided. In this article we solve this age old verification problem by introducing a novel open source verification language called Vlang [1]. Vlang is built on top of D Programming Language [2], and consequently it inherits support for parallel, generic, generative, and functional programming paradigms in addition to Object Oriented Programming. Vlang is ABI compatible with C/C++, creates single executable with SystemC [3], [4] and integrates with System Verilog [5] seamlessly through VPI/DPI. Currently Vlang supports UVM-1.1d standard as in built methodology library.

Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf (DVCon Europe 2014)

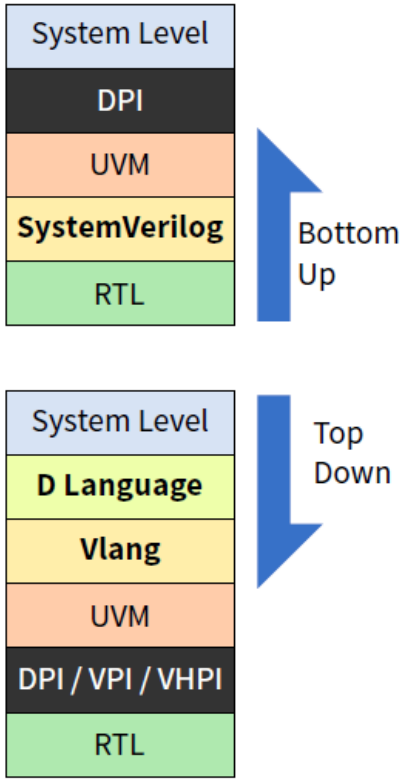
Top Down Verification Stack

System Verilog integrates tightly with RTL;
Vlang with System Level

- Vlang is built on top of D Programming Language which provides ABI compatibility with C/C++
- Vlang interfaces with RTL using DPI
  - DPI overhead is compensated by parallel execution of testbench
- **Vlang offers zero communication overhead when integrating with Emulation Platforms and with Virtual Platforms**

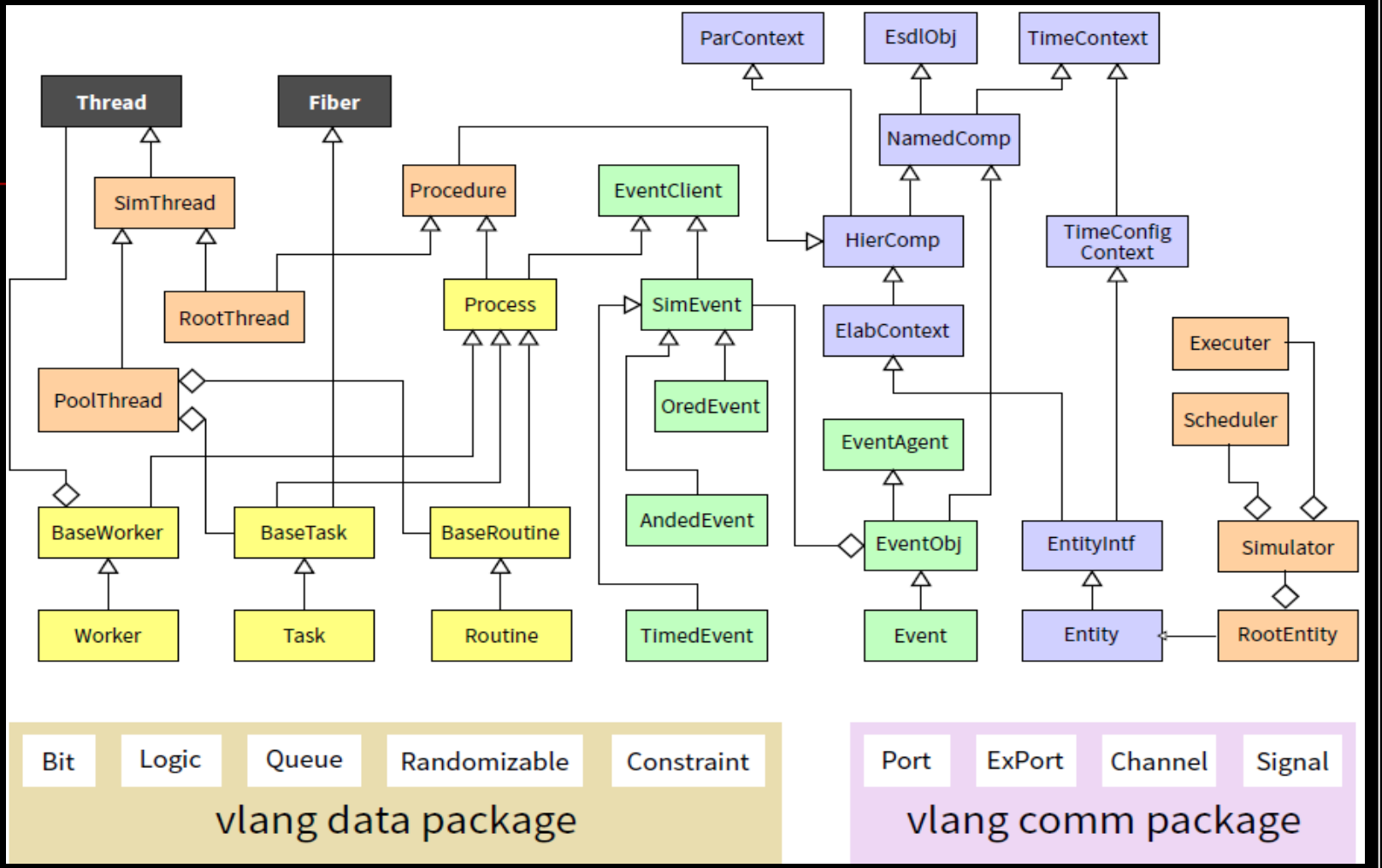


Vlang Cosimulation with Virtual/Emulation Platforms



Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf (DVCon Europe 2014)

Core Infrastructure



Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf (DVCon Europe 2014)

Processes

- Processes and Forks (and for that matter everything else) in Vlang is an object
  - You can pass an Event, a Process, or a Fork as an argument to a function
- Joining a fork can be done flexibly with the Fork object
- Forks and Processes can be suspended, disabled, aborted etc

```
void frop() {
  Fork zoo = fork
    ({ // fork1
       foo();
     },
     { // fork2
       bar();
     }).joinAny();
  // Some Code
  zoo.join();
  zoo.abortTree();
}
void foo() {
  auto proc = Process.self();
  proc.abortTree();
}
void bar() {
  // wait for fork branch
  Fork ff = Fork.self();
  ff.joinAny();
  // ....
}
```

Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf

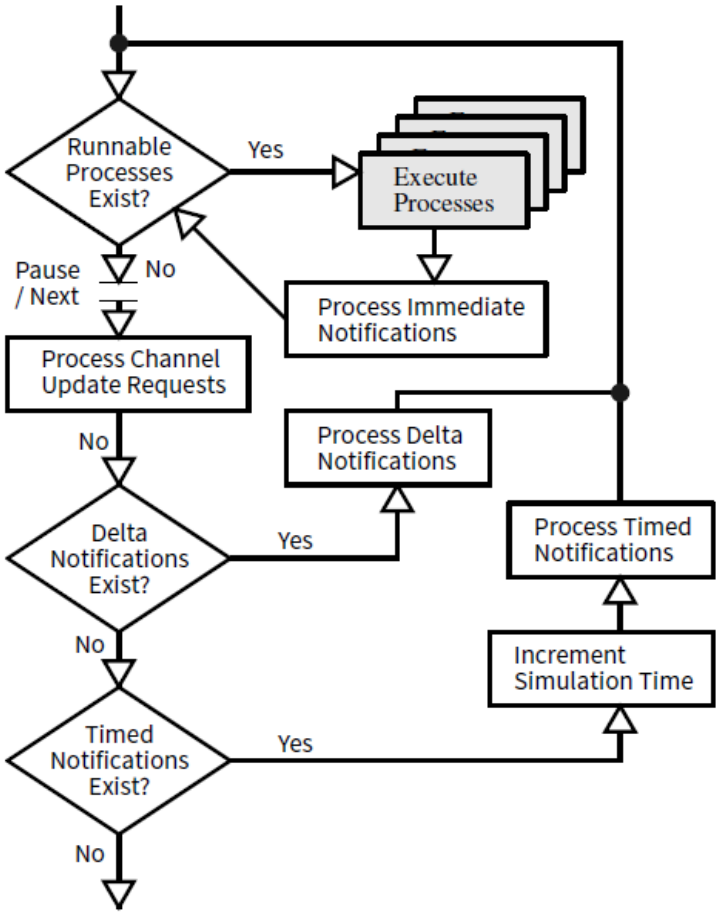
Constrained Randomization

- System Verilog style *Constrained Randomization*
- Class needs to be derived from `Randomizable`
- **`mixin Randomization` magic makes `randomize` polymorphic**
- Within UVM (and that is what matters) all the ugliness is gone
- Support for common arithmetic, logical and comparison operators
- Support for array and dynamic array randomization
- Support for `if-else`, `foreach`
  - Can be freely nested

```
class Foo: Randomizable {
  mixin Randomization;
  @rand!8 byte[] foo;
  @rand Logic!12 baz;
}
class Bar: Foo {
  mixin Randomization;
  @rand ubyte[8] bar;
  Constraint! q{
    foo.length > 2; // array
    baz[0..8] == 16;
  } cstFooLength;
  Constraint! q{
    foreach(i, f; bar) f <= i;
    foreach(i, f; foo) {
      if(i > 4) /*condition*/
        f + i < 32 && f > 16;
    }
  } cstFoo;
}
void main() {
  Foo randObj = new Bar();
  for (size_t i=0; i!=10; ++i) {
    randObj.randomize();
  }
}
```

Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf (DVCon Europe 2014)

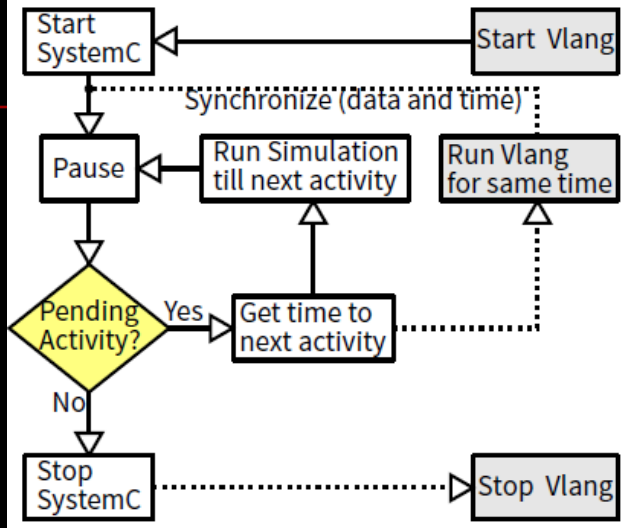
Multicore UVM



- Vlang simulator is multicore capable
- Vlang implementation of Multicore UVM takes advantage of the fact that there is minimal interaction between different `uvm_agents`
- Vlang provides an abstraction `ParContext` to manage parallelization
- The uvm constructs (like `uvm_objection`), that are shared between the components, are *synchronized* in the UVM base library implementation
  - In Vlang port of UVM, a `uvm_component` implements `ParContext`

Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf (DVCon Europe 2014)

Interfacing with SystemVerilog and SystemC



- Vlang simulator can be fully synchronized with SystemC and SystemVerilog

- With Systemc, Vlang can lock at delta cycle level

```
int main(int argc, char* argv[]) {
  initEsdl();          // initialize vlang
  int screslut =
    sc_core::sc_elab_and_sim(argc, argv);
  finalizeEsdl();      // stop vlang
  return 0;
}
int sc_main( int argc, char* argv[]) {
  sc_set_time_resolution(1, SC_PS);
  top = new SYSTEM("top");
  sc_start( SC_ZERO_TIME );
  while(sc_pending_activity()) {
    sc_core::sc_time time_ =
      sc_time_to_pending_activity();
    // start vlang simulation for given time
    esdlStartSimFor(time_.value());
    sc_start(time_);
    // wait for vlang to complete time step
    esdlWait();
  }
  return 0;
}
```

Source: https://dvcon-proceedings.org/wp-content/uploads/introduction-to-next-generation-verification-language-vlang-presentation.pdf (DVCon Europe 2014)

Comparison

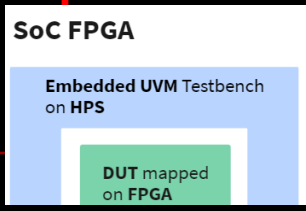
- **https://fdocuments.in/document/vlang-flyer.html?page=2**
  (Date Post 28-Dec-2015)

Feature	Vlang	System-Verilog	SystemC	Remarks
Performance Enablers				
Concurrent Threads	Yes	No	No	Use -version=MULTICORE to enable
Multiple Concurrent Simulators	Yes	No	No	
Native Compilation	Yes	No	Yes	
Generic Library Support	Yes	No	Yes	No standard library for SV
Coding Productivity				
Compile Time	Fastest	Slow	Fast	
Incremental Compile	Yes	Partial	Yes	Huge elaboration time for SV
Pointer-less Programming	Yes	Yes	No	
Automatic Garbage Collection	Yes	Yes	No	Every significant language born after year 2000 provides for a decent GC
User-friendly Containers	Yes	Yes	No	Associative arrays vs C++`std::map<>` `[][]` vs multi-dimensional vectors in C++
Runtime Safety				
Array Bound Check	Yes	No	No	
Check for Integral Overflow	Yes	No	No	
Support for Unittests	builtin	library	library	Vlang supports localized Unittests.
Exception Handling	Yes	No	Yes	
Contract-based Programming	Yes	No	No	

Systems Programming Features				
Low level hardware/device access	Yes	No	Yes	Essential for Virtual Platforms and Coverification
Custom memory allocation	Yes	No	Yes	
Efficient File IO	Yes	No	Yes	
Parsing tools/libraries	Yes	No	Yes	SV can not parse even XML by itself
Embedded Assembly Code	Yes	No	Yes	
Reflections and Generative Programming				
Support for Data Introspection and Reflections	Yes	No	No	Very useful for UVM automation
Generative and Metaprogramming Support	Yes	No	Limited	
UVM Support				
Base Class Libraries	Yes	Yes	Yes	
Support for Sequences	Yes	Yes	Limited	SystemC lacks `randomize with`
Register Abstraction Layer	No	Yes	No	RAL package for Vlang is under development
TLM1 Support	Yes	Yes	Yes	
TLM2 Support	No	Yes	Yes	TLM2 support in SystemVerilog is limited Support for Vlang TLM2 in the works
Verification Features				
Transaction Randomization	Yes	Yes	Limited	SystemC relies on external constraint libraries, none of which is user friendly
Sequence Randomization	Yes	Yes	No	
Coverage Support	No	Yes	No	Coverage will be available in next release of Vlang

UVM Architecture Overview

- http://uvm.io/

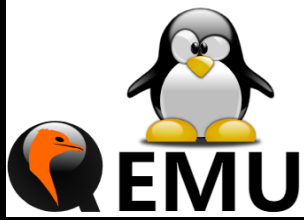


Spawn your own Emulation Platform for $100

Create your own SoCFPGA based Emulator for $100 and upto 100X speedup, with an **Embedded UVM** testbench running on HPS and DUT mapped on FPGA.

Run UVM Tests with Vivado and with GHDL

Opensource and Free **IEEE UVM 1.0** port complete with Constrained Reandomization, released under Apache2/Boost license.

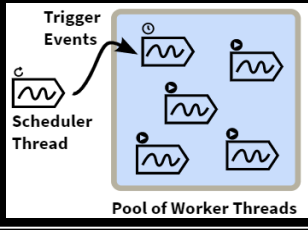


Co-Simulate your DUT with Device Drivers

LLVM powered native compilation on ARM and other embedded processors, with runtime Footprint small enough to run UVM on Raspberry PI and Beaglebone.

Deploy UVM Testbenches on Software Stack

LLVM powered native compilation on ARM and other embedded processors, with runtime Footprint small enough to run UVM on Raspberry PI and Beaglebone.

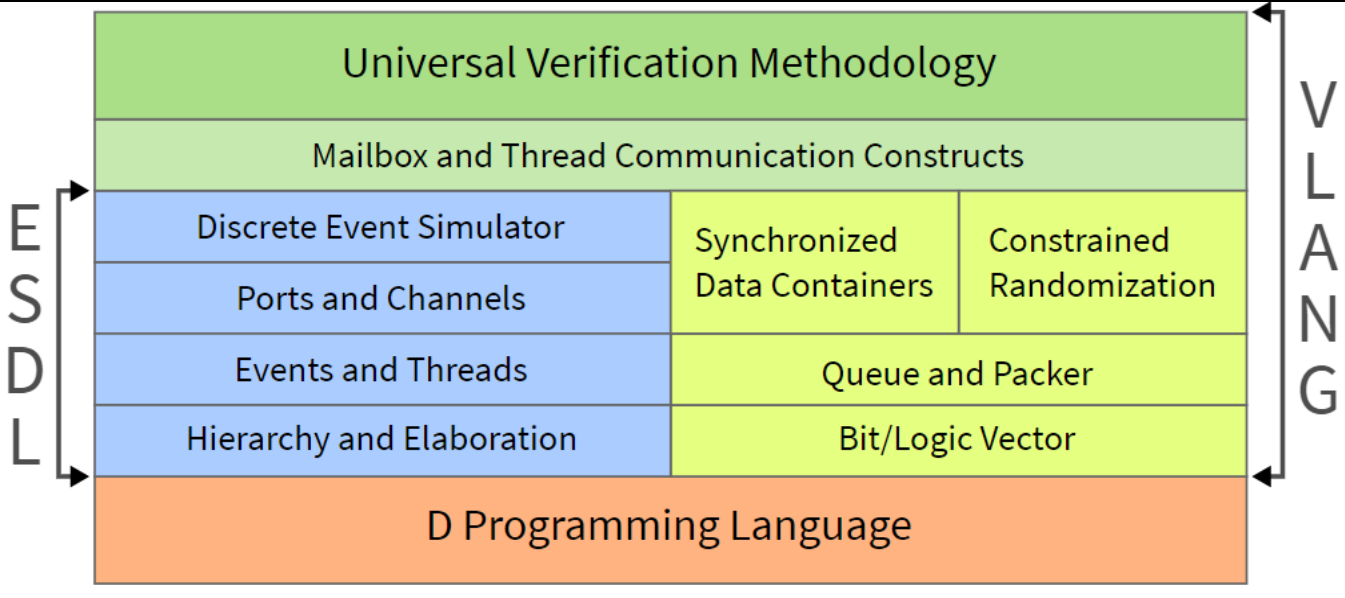


Scale your Testbench to Multicore Servers

The first and yet the only UVM implementation that lets your testbench run on multiple cores. Lets your testbench scale on Multicore server machine.

Vlang is built on top of the D Programming Language (Why D?). At the core of Vlang is a system specification language called **Electronic System Description Language** (ESDL). ESDL is much like SystemC, but is written from scratch in D Language and has many improvisations including ability to run simulation on a multicore system.

is a Discrete Event Simulator
and the associated constructs. Vlang also defines hardware data types that make it convenient for the verification engineer to model bit/logic vectors and signals. Also included in the core is a BDD based constraint solver and a glue library that allows constraints to be declared as part of a data transaction object. On top of the core layer, Vlang implements a port of the Universal Verification Methodology (UVM).



Source: http://uvm.io/docs/core-concepts/architecture-overview/

Design Elaboration

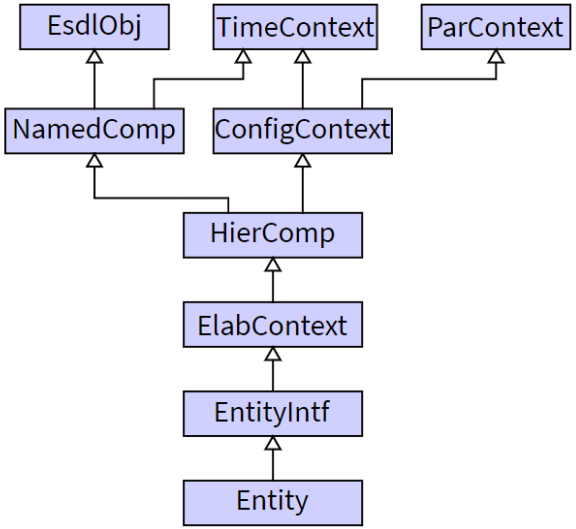
Elaboration in ESDL happens in phases. A component in ESDL is called an Entity. The diagram below shows the base class hierarchy of an Entity. To make elaboration of a design possible, each Entity in the design must have auxiliary code that reflects on the composition of the component. This is done by adding the following line as part of the Entity class declaration:

```
mixin Elaboration;
```

That is it. Normally you will not be required to add any other line of code in the entity to facilitate its elaboration. The Elaboration mixin would add the required code to the Entity. For example the code added by the mixin would:

- create a build function that will construct (new) all the component entities declared in the body of the current entity
- call build function for those component entities as well.

An ESDL *Entity* is analogous to `module` in SystemVerilog or an `sc_module` in SystemC. ESDL chose to use Entity because `module` is a keyword in D Programming Language.



Source: http://uvm.io/docs/core-concepts/hierarchy/

Notion of Time

Just like SystemVerilog, time comes in ESDL in two forms. You can specify time in absolute form complete with a unit. In this form, the value of time is stored as a 64-bit value and the unit is stored as a byte enum. Since they are clubbed together in form of a structure, Time specified in this form takes 128 bits (16 bytes) of data.

More often, you would want to look at time as an integral value. ESDL uses a wrapper struct named SimTime for such integral modeling of time. SimTime gets the absolute sense only in conjunction with time precision used by the simulation. Since ESDL allows multiple simulator instances running on parallel threads, time precision is tagged with a simulator instance.

Source: http://uvm.io/docs/core-concepts/hierarchy/

2.2.2 ESDL

- https://github.com/coverify/esdl

Electronic System Description Language.

This package has the base simulator and the data modules, which include the constrained randomization unit. UVM package is available separately because it comes with a different opensource license.

- Src

Stats (master branch and last commit: ee6f653236c6ed2a461433c25899f54f9aa39a24)

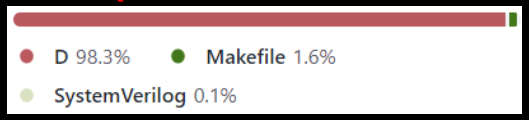
```
[mydev@fedora esdl]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 C Header                3         3512         1173         2129          210
 C++                     2          127           64           41           22
 D                     132        88411        56872        20229        11310
 JSON                    2           33           33            0            0
 Markdown                1           12            0            8            4
===============================================================================
 Total                 140        92095        58142        22407        11546
===============================================================================
[mydev@fedora esdl]$
```

```
[mydev@fedora esdl]$ tree -L 5 -d .
.
├── examples
│   ├── data
│   ├── rand
│   │   └── misc
│   ├── sim
│   └── vcd
└── src
    ├── esdl
    │   ├── base
    │   ├── data
    │   ├── intf
    │   │   ├── btor
    │   │   ├── verilator
    │   │   │   └── cpp
    │   │   └── z3
    │   │       └── api
    │   ├── posix
    │   │   └── sys
    │   │       └── net
    │   ├── rand
    │   ├── solver
    │   ├── sync
    │   ├── sys
    │   └── vcd
    └── unstd
        └── memory
```

2.2.3 EUVM

- https://github.com/coverify/euvm
  Embedded UVM (depends on ESDL)
  D port of IEEE standard 1800.2 2020-1.0 UVM (Universal Verification Methodology).
- Src

Stats (master branch and last commit: f5aaca012ba08d3a1ec018018337f0bd821da285)




```
[mydev@fedora euvm]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 D                     200        86907        47034        26934        12939
 JSON                    2           42           42            0            0
 Makefile               37         1343          401          653          289
 Markdown                1            5            0            4            1
 SystemVerilog           1           65           51            3           11
===============================================================================
 Total                 241        88362        47528        27594        13240
===============================================================================
[mydev@fedora euvm]$
```

```
[mydev@fedora euvm]$ tree -L 3 .
.
├── dub.json
├── dub.selections.json
├── LICENSE
├── README.md
├── src
│   ├── uvm
│   │   ├── base
│   │   ├── comps
│   │   ├── dap
│   │   ├── dpi
│   │   ├── meta
│   │   ├── package.d
│   │   ├── reg
│   │   ├── seq
│   │   ├── tlm1
│   │   ├── tlm2
│   │   └── vpi
│   └── uvm_pkg.d
├── tests
│   ├── 00basic
│   │   ├── 00hello
│   │   ├── 01compfail
│   │   ├── 02runfail
│   │   ├── 03error
│   │   ├── 06plusargs
│   │   ├── 07toolargs
│   │   ├── 10post_test
│   │   ├── 20subgroup
│   │   ├── 25typename
│   │   └── 90Mantis
│   ├── 01report
│   │   ├── 00message
│   │   ├── 02server
│   │   ├── 10catcher
│   │   ├── 20severity
│   │   ├── 30handler
│   │   └── 40macros
│   ├── 04vlang
│   │   └── 00set_local
│   └── test_defines.mk
└── verilog
    └── uvm_dpi_utils.sv
```

2.2.4 Summary

- ESDL-EUVM only support UVM 1.0;
- The code of ESDL-EUVM at https://github.com/coverify/ lacks of documents and test result;
- The latest binary release EUVM v1.0-beta25 targets X64 only:
  https://github.com/coverify/euvm/releases


```
[mydev@fedora euvm-1.0-beta25]$ file lib/libuvm-ldc-shared.so.0.0.2
lib/libuvm-ldc-shared.so.0.0.2: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, BuildID[sha1]=6a
305cb6c939de04bc7de1473d21808b3d311ba2, not stripped
[mydev@fedora euvm-1.0-beta25]$ file lib/libesdl-ldc-shared.so.0.0.2
lib/libesdl-ldc-shared.so.0.0.2: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, BuildID[sha1]=b
140692473dd80de8003680cf8e1865db26bed01, not stripped
[mydev@fedora euvm-1.0-beta25]$
```

The license file within it:

```
[mydev@fedora euvm-1.0-beta25]$ cat ./LICENSE.txt
The following license applies to Software that is:

1. the executables in the bin directory
2. the libraries in the lib directory
3. the supplied source code

LDC, the D compiler is covered under the license file LICENSE_LDC.txt.
Z3, an SMT Solver by Microsoft is covered under the license file LICENSE_Z3.txt.
Embedded UVM port of IEEE UVM 2017-1.0 Reference Implementation is covered under the license file LICENSE_UVM.txt

The ESDL part of the package is covered under Boost Software License - Version 1.0.
```

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
[mydev@fedora euvm-1.0-beta25]$
```

- Current code of ESDL-EUVM failed to run on RPi4;
- Working on workarounds and trying to make ESDL-EUVM work on RPi4 with upstream D;
- Anyway, ESDL-EUVM provides good reference design and code base for us.

III. D for IR

2) GyreD

2.1 Overview

- https://github.com/baioc/gyred

  A new compiler IR (WIP).

> This is a reference implementation of Gyre, a compiler IR designed as a practical "linguistic switchbox", or UNCOL.
>
> As such, the IR should be able to efficiently accommodate many high-level languages and target architectures. In order to test this, this repository ~~contains~~ will contain multiple compilers using the same middle-end infrastructure.

UNCOL

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

UNCOL (**Universal Computer Oriented Language**) is a universal intermediate language for compilers. The idea was introduced in 1958, by a SHARE ad-hoc committee.[1] It was never fully specified or implemented; in many ways it was more a concept than a language.

UNCOL was intended to make compilers economically available for each new instruction set architecture and programming language, thereby reducing an N×M problem to N+M.[2] Each machine architecture would require just one compiler back end, and each programming language would require one compiler front end. This was a very ambitious goal because compiler technology was in its infancy, and little was standardized in computer hardware and software.

History [edit]

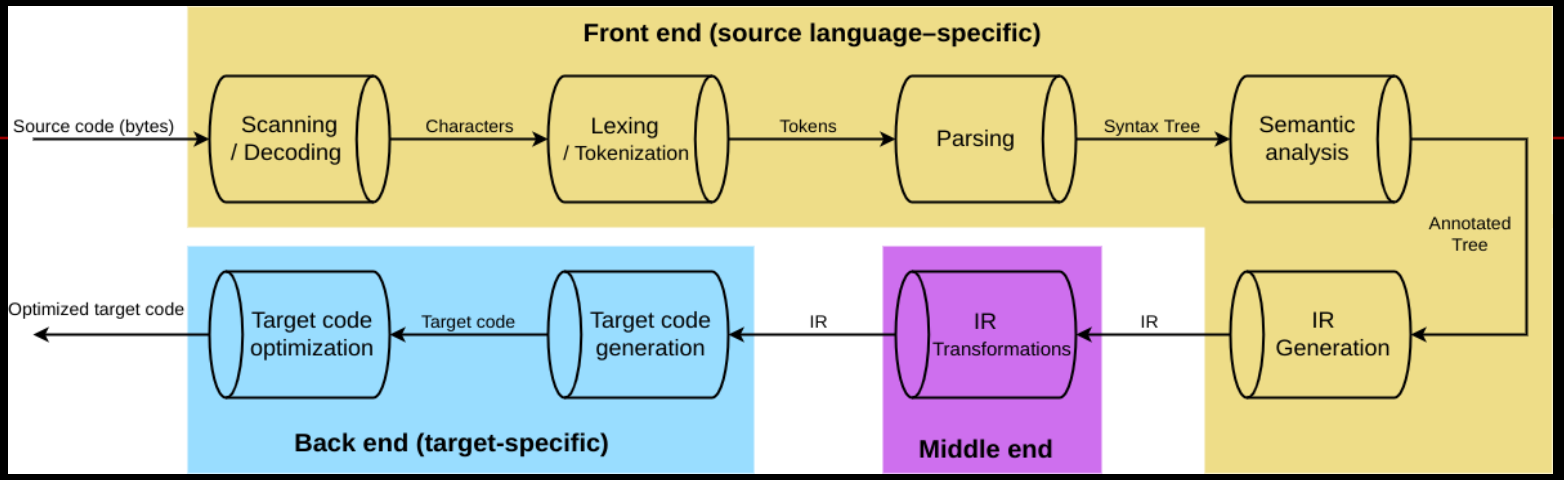
The concept of such a universal intermediate language is old: the SHARE report (1958) already says "[it has] been discussed by many independent persons as long ago as 1954." Macrakis (1993) summarizes its fate:

> UNCOL was an ambitious effort for the early 1960s. An attempt to solve the compiler-writing problem, it ultimately failed because language and compiler technology were not yet mature. In the 1970s, compiler-compilers ultimately contributed to solving the problem that UNCOL set itself: the economical production of compilers for new languages and new machines.

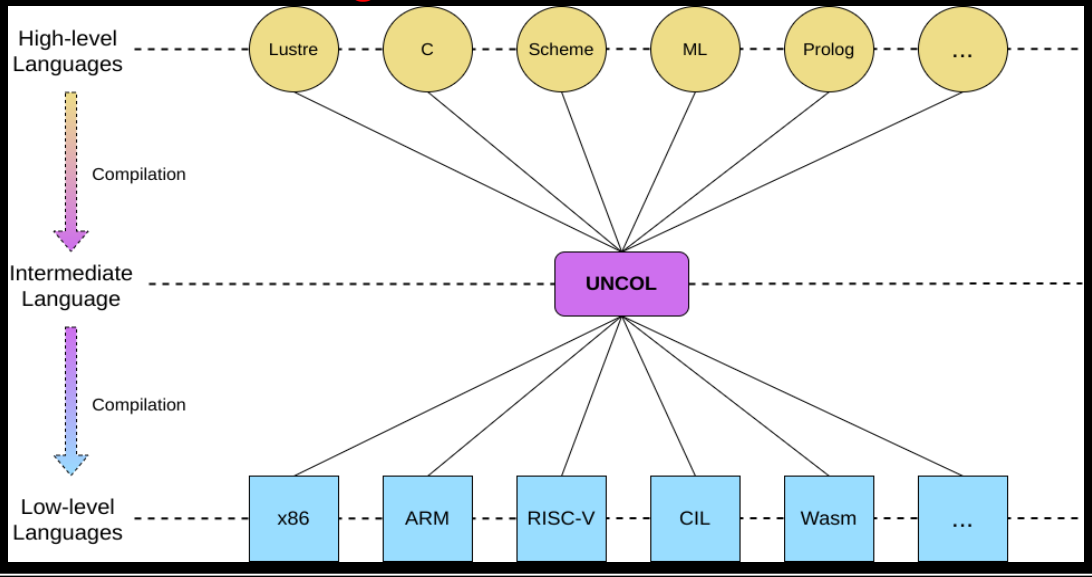
UNCOL is sometimes used as a generic term for the idea of a universal intermediate language. The Architecture Neutral Distribution Format is an example of an UNCOL in this sense, as are various bytecode systems such as UCSD Pascal's p-code, and most notably Java bytecode.[3]

Current design

- Logical structure of an optimizing compiler



- UNCOL as a linguistic switchbox



- **https://baioc.github.io/gyred/gyre.html**

Gyre is not particularly innovative, aiming instead to consolidate various existing techniques under a unified design. It's a graph-based IR, heavily influenced by Click's sea of nodes, although we take a more functional approach to SSA form by mixing Extended Basic Blocks with Thorin-style-CPS. Explicit concurrency comes with the addition of a standard memory consistency model and a message-passing mechanism from the join calculus. Several other features are "mashed up" in Gyre's design, coming from an extensive survey on compilers, portable IRs and the UNCOL solution to the "compiler construction problem".

> "The language designer should be familiar with many alternative features designed by others ... One thing he should not do is to include untried ideas of his own. His task is consolidation, not innovation." - Tony Hoare, 1989

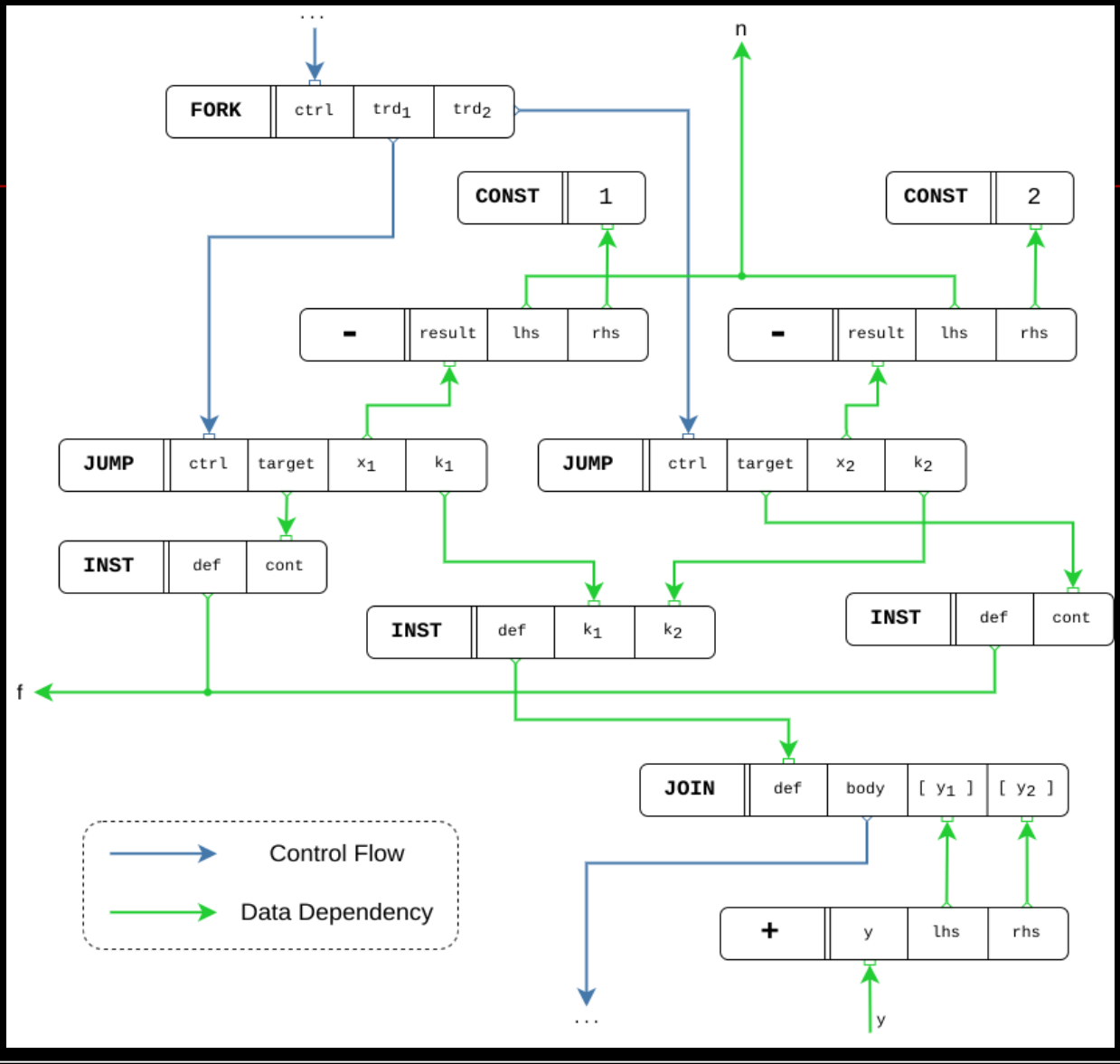
...

- **Design Principles**

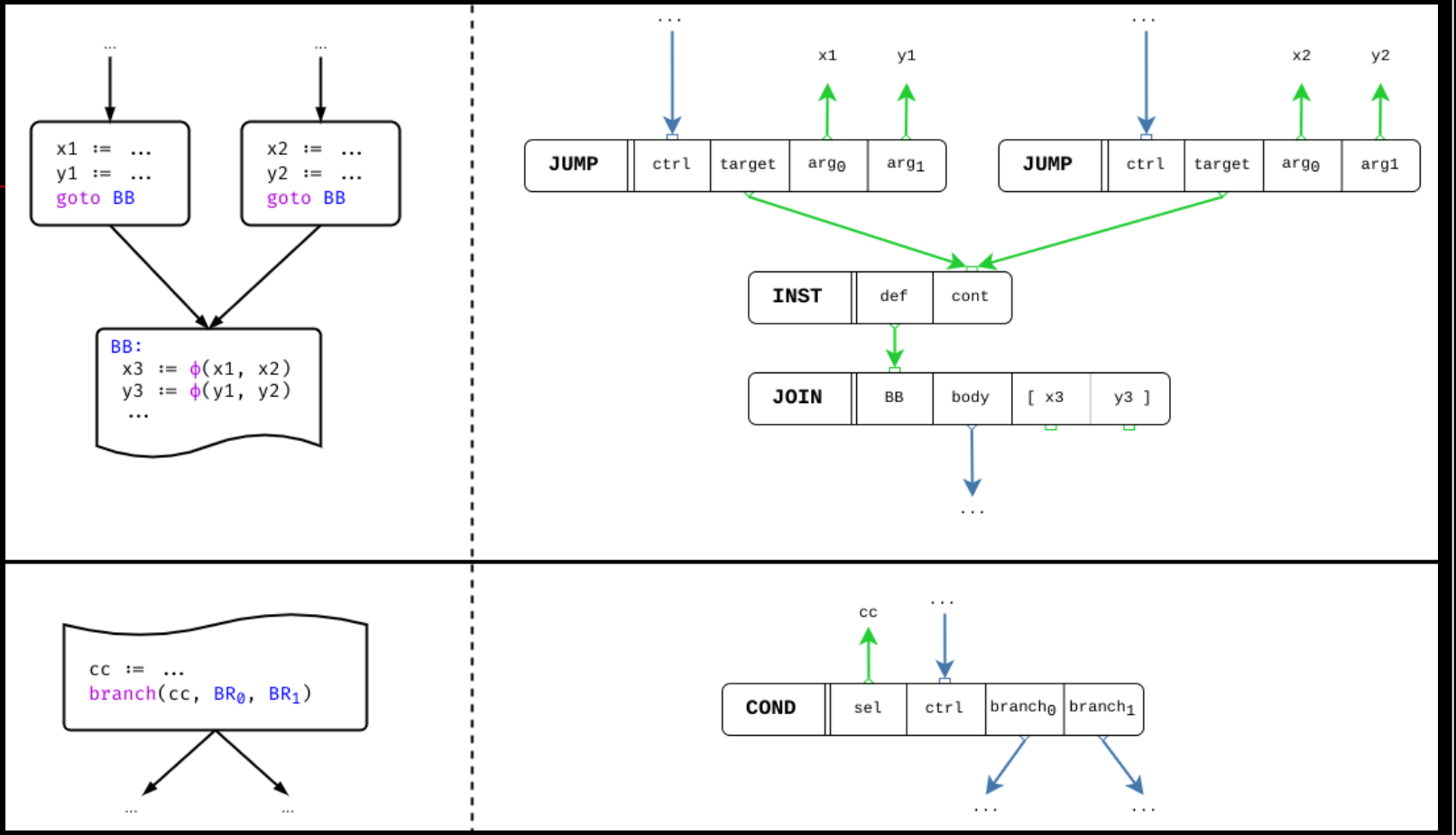
- Expressivity: The IR must be able to represent, up to some level of detail, the essential properties of any program. Software which is not portable is also of interest, even if only to indicate where and how it uses machine-specific constructs and assumptions, or when compiling it to a compatible architecture (something which ought to be supported).
- Efficiency: The IL format should be fast to decode and validate. At the same time, the IR itself must not introduce unreasonable costs to code transformations, nor impose additional restrictions (unnecessary from a logical standpoint) to its programs.
- Progressivity: An optimizing compiler should prefer to retain, rather than recover, in- formation about a program. Premature lowering decreases optimization potential, so the IR must be able to preserve "high-level" structure for as long as it is desirable to do so.
- Parsimony: The IR must not expand uncontrollably to accommodate new languages or machines. Instead, it ought to contain a well-defined core which is simple, yet expressive.
- Extensibility: The IL should be amenable to extensions, both from users and to its own specification. These must be clearly signaled (possibly at different levels of program granularity) for the sake of compatibility. This principle complements the previous one.

- ...

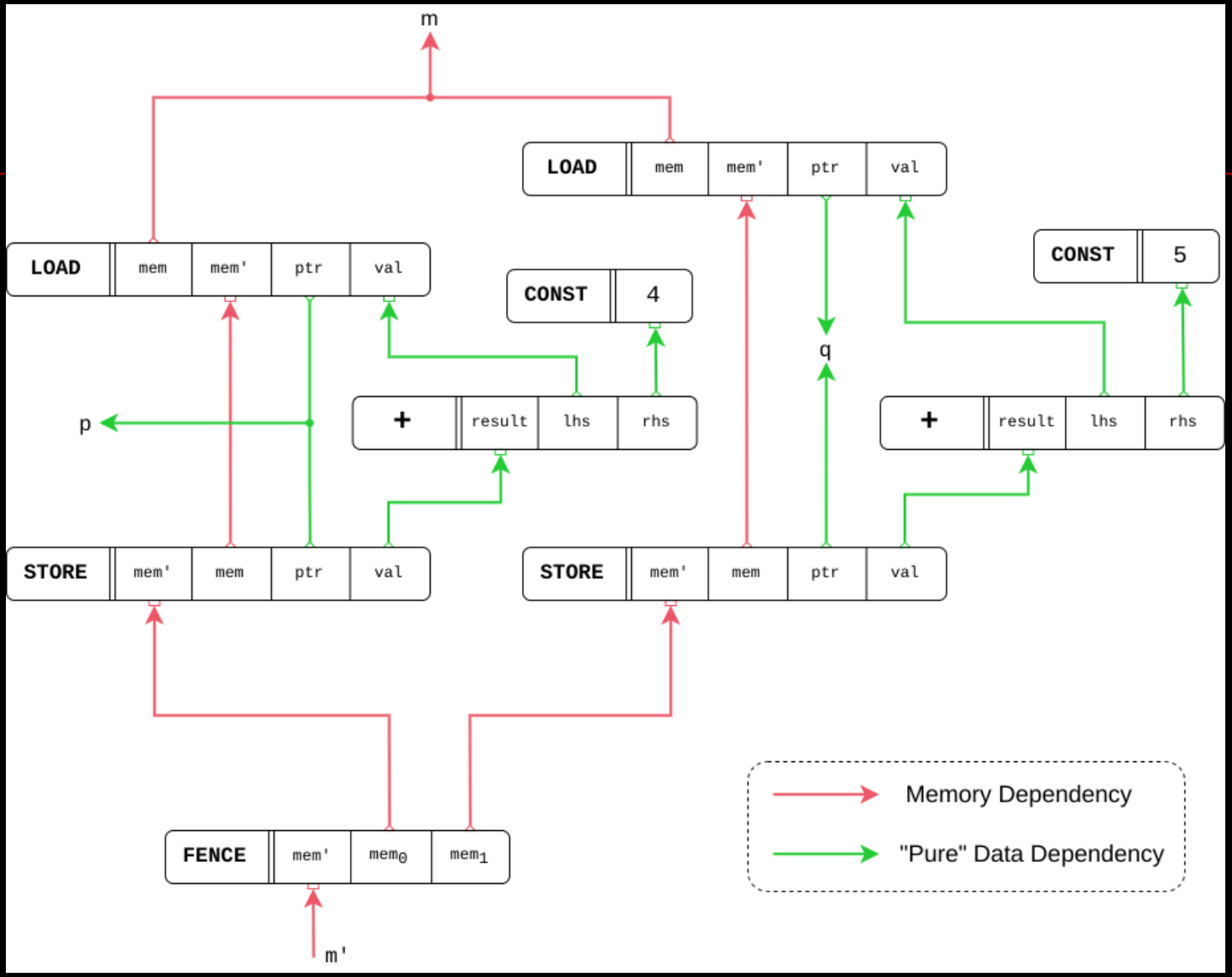
Gyre subgraph with explicit concurrency



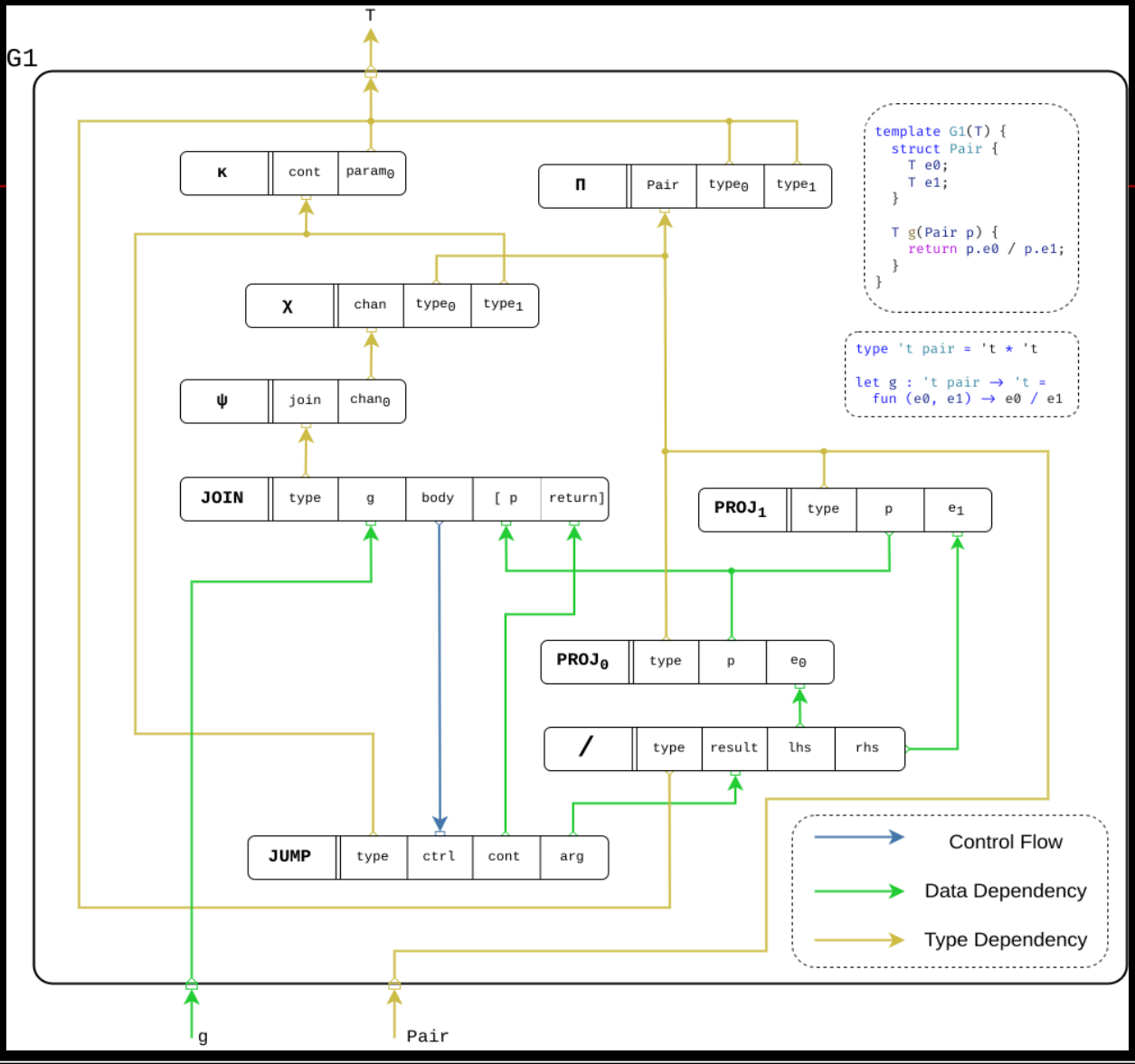
Mapping of classic SSACFG constructs



Memory operations in Gyre



IR types and macros



Src

- **Stats**

  master branch with last commit: 0f6d6c6c1a6be9b78687cfac022ca547ac9c6404

```
[mydev@fedora gyred]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 D                      10         7016         4352         1770          894
 JSON                    1           42           37            0            5
 Plain Text              1          201            0          151           50
-------------------------------------------------------------------------------
 Markdown                1           54            0           35           19
 ├─ JSON                 1           73           73            0            0
 (Total)                             127           73           35           19
===============================================================================
 Total                  13         7313         4389         1956          968
===============================================================================
```

```
[mydev@fedora gyred]$ tree
.
├── dub.json
├── LICENSE.txt
├── README.md
└── source
    ├── app.d
    ├── eris
    │   ├── core.d
    │   ├── hash_table.d
    │   ├── package.d
    │   ├── rational.d
    │   └── util.d
    └── gyre
        ├── graph.d
        ├── mnemonics.d
        ├── nodes.d
        └── package.d
```

- **dependencies**

  betterclist: https://github.com/gilzoide/betterclist

2) idjit

2.1 Overview

- https://github.com/BradleyChatha/idjit

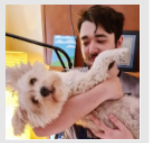
A barebones project for playing around with DMD-FE and MIR.

```
[mydev@fedora idjit]$ tree -L 2 .
.
├── dmd.meson.build
├── meson.build
├── mir.meson.build
├── README.md
├── setup.lua
├── source
│   └── idjit
└── subprojects
    ├── dmd
    └── mir
```

```
[mydev@fedora idjit]$ tree source/idjit/
source/idjit/
└── example.d
```

https://forum.dlang.org/post/vhfyhlhahqahlodevddv@forum.dlang.org

SealabJaster



Posted in reply to Menshikov

On Friday, 12 November 2021 at 13:36:35 UTC, Menshikov wrote:

> I've only seen SDC. Maybe there are some other options?

I've been wondering for a while now how viable it would be to make a subset of D for the MIR JIT library: https://github.com/vnmakarov/mir

It'd be interesting to say the least: "Emebedable D"

- https://github.com/BradleyChatha/idjit/blob/master/source/idjit/example.d

Failed to build on RPi4

- 1. build dstep(https://github.com/jacob-carlborg/dstep) firstly
- 2.

```
[mydev@fedora idjit]$ export PATH=$PATH:/opt/MyWorkSpace/MyProjs/Languages/D/Compilation/DStep/Official/dstep/bin
[mydev@fedora idjit]$ which dstep
/opt/MyWorkSpace/MyProjs/Languages/D/Compilation/DStep/Official/dstep/bin/dstep
[mydev@fedora idjit]$ file /opt/MyWorkSpace/MyProjs/Languages/D/Compilation/DStep/Official/dstep/bin/dstep
/opt/MyWorkSpace/MyProjs/Languages/D/Compilation/DStep/Official/dstep/bin/dstep: ELF 64-bit LSB executable, ARM aarch64, versi
on 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=489d55e082d78e3d6468d3a63057599a6077017
f, for GNU/Linux 3.7.0, with debug_info, not stripped
[mydev@fedora idjit]$
[mydev@fedora idjit]$ meson build
The Meson build system
Version: 0.62.2
Source dir: /opt/MyWorkSpace/MyProjs/Languages/D/Runtime/MIR/IDJIT/Official/idjit
Build dir: /opt/MyWorkSpace/MyProjs/Languages/D/Runtime/MIR/IDJIT/Official/idjit/build
Build type: native build
Project name: idjit
Project version: undefined
C compiler for the host machine: ccache cc (gcc 12.2.1 "cc (GCC) 12.2.1 20220819 (Red Hat 12.2.1-1)")
C linker for the host machine: cc ld.bfd 2.37-27
D compiler for the host machine: ldc2 (llvm 1.30.0 "LDC - the LLVM D compiler (1.30.0):")
D linker for the host machine: ldc2 ld.bfd 2.37-27
Host machine cpu family: aarch64
Host machine cpu: aarch64
WARNING: You should add the boolean check kwarg to the run_command call.
         It currently defaults to false,
         but it will default to true in future releases of meson.
         See also: https://github.com/mesonbuild/meson/issues/9300

meson.build:8:0: ERROR: Subproject exists but has no meson.build file

A full log can be found at /opt/MyWorkSpace/MyProjs/Languages/D/Runtime/MIR/IDJIT/Official/idjit/build/meson-logs/meson-log.tx
t
[mydev@fedora idjit]$
```

and meson-log.txt shows this failure is caused by “lumarsh”

https://github.com/BradleyChatha/lumarsh

```
[mydev@fedora idjit]$ grep -ir "lumarsh" .
./meson.build:run_command('dub', 'run', 'lumarsh', '--', './setup.lua', 'dmd')
./meson.build:run_command('dub', 'run', 'lumarsh', '--', './setup.lua', 'mir')
./setup.lua:if #LUMARSH_ARGS > 0 and LUMARSH_ARGS[1] == 'dmd' then
./setup.lua:elseif #LUMARSH_ARGS > 0 and LUMARSH_ARGS[1] == 'mir' then
```

- Firstly, let's clarify the dependencies:

  lumarsh depends on lumars(https://github.com/BradleyChatha/lumars) and jcli (https://github.com/BradleyChatha/jcli);
  lumars depends on bindbc-lua(https://github.com/BindBC/bindbc-lua) and taggedalgebraic(https://github.com/s-ludwig/taggedalgebraic)

- Try to build lumars and lumarsh manually with the latest version of every dependency on RPi4 and then:

  prepare dynamic and static library for Lua 5.1 for the need of lumars (including download Lua 5.1 static library for AArch64 from https://centos.pkgs.org/7/centos-aarch64/lua-static-5.1.4-15.el7.aarch64.rpm.html)

```
[mydev@fedora lumars]$ ll /usr/lib64 |grep -i lua
-rw-r--r--. 1 mydev mydev    356484 Nov 23  2016 liblua-5.1.a
-rwxr-xr-x. 1 root  root     270624 Jan 19  2022 liblua-5.1.so*
-rwxr-xr-x. 1 root  root     271592 Jul 19 07:57 liblua-5.3.so*
-rwxr-xr-x. 1 root  root     337720 Jul 19 07:57 liblua-5.4.so*
-rw-r--r--. 1 root  root     850098 Jul 19 07:57 liblua.a
lrwxrwxrwx. 1 root  root         22 Aug 24 23:20 libluajit-5.1.so → libluajit-5.1.so.2.1.0*
lrwxrwxrwx. 1 root  root         22 Aug 24 23:20 libluajit-5.1.so.2 → libluajit-5.1.so.2.1.0*
-rwxr-xr-x. 1 root  root     607744 Aug 24 23:20 libluajit-5.1.so.2.1.0*
lrwxrwxrwx. 1 root  root         20 Jan 20  2022 libtexlua53.so.5 → libtexlua53.so.5.3.6*
-rwxr-xr-x. 1 root  root     270752 Jan 20  2022 libtexlua53.so.5.3.6*
lrwxrwxrwx. 1 root  root         21 Jan 20  2022 libtexluajit.so.2 → libtexluajit.so.2.1.0*
-rwxr-xr-x. 1 root  root     605672 Jan 20  2022 libtexluajit.so.2.1.0*
drwxr-xr-x. 1 root  root         18 Jul 19 07:57 lua/
drwxr-xr-x. 1 root  root          6 Mar 28 12:01 luajit/
[mydev@fedora lumars]$
```

sucessfully build lumars with the following patch:

```
[mydev@fedora lumars]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   dub.sdl

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        deps/aarch64/

no changes added to commit (use "git add" and/or "git commit -a")
[mydev@fedora lumars]$
[mydev@fedora lumars]$ git diff
diff --git a/dub.sdl b/dub.sdl
index 8e5061e..680f04e 100644
--- a/dub.sdl
+++ b/dub.sdl
@@ -8,6 +8,7 @@ dependency "taggedalgebraic" version="~>0.11.22"
 configuration "lua51" {
        targetType "library"
        dflags "-L$PACKAGE_DIR/deps/linux64/lua51.a" platform="linux-x86_64"
+       dflags "-L$PACKAGE_DIR/deps/aarch64/linux/liblua-5.1.a" platform="linux-aarch64"
        dflags "-L$PACKAGE_DIR/deps/win64/lua51.lib" platform="windows-x86_64"
        dflags "-L$PACKAGE_DIR/deps/macamd/lua51.a" platform="osx-aarch64"
        versions "BindLua_Static" "LUA_51"
@@ -15,6 +16,7 @@ configuration "lua51" {
 }
 configuration "lua51-dynamic" {
        targetType "library"
+       copyFiles "$PACKAGE_DIR/deps/aarch64/linux/liblua-5.1.so" platform="linux-aarch64"
        copyFiles "$PACKAGE_DIR/deps/win64/lua51.dll" platform="windows-x86_64"
        copyFiles "$PACKAGE_DIR/deps/macamd/liblua.5.1.dylib" platform="osx-aarch64"
        versions "LUA_51"
[mydev@fedora lumars]$
```

```
[mydev@fedora lumars]$ tree deps/aarch64/
deps/aarch64/
└── linux
    ├── liblua-5.1.a
    └── liblua-5.1.so
```

```
[mydev@fedora lumars]$ dub build -b release
Performing "release" build using /opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 for aarch64, arm_hardfloat.
lumars 1.5.0+commit.1.ga12c005: building configuration "lua51"...
[mydev@fedora lumars]$
```

and successfully build lumarsh with the patch as below:

```
[mydev@fedora lumarsh]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   dub.sdl
        modified:   dub.selections.json

no changes added to commit (use "git add" and/or "git commit -a")
[mydev@fedora lumarsh]$ git diff
diff --git a/dub.sdl b/dub.sdl
index 2b41174..dc3ff7a 100644
--- a/dub.sdl
+++ b/dub.sdl
@@ -3,5 +3,5 @@ description "Use LUA for shell scripting"
 authors "Bradley Chatha"
 copyright "Copyright © 2021, Bradley Chatha"
 license "MIT"
-dependency "lumars" version="~>1.1.6"
-dependency "jcli" version="~>0.22.4"
+dependency "lumars" version="~>1.5.0"
+dependency "jcli" version="~>0.24.0"
diff --git a/dub.selections.json b/dub.selections.json
index f53bf4e..6be377c 100644
--- a/dub.selections.json
+++ b/dub.selections.json
@@ -1,10 +1,10 @@
 {
        "fileVersion": 1,
        "versions": {
-               "bindbc-loader": "0.3.2",
-               "bindbc-lua": "0.4.1",
-               "jcli": "0.22.4",
-               "lumars": "1.1.6",
+               "bindbc-loader": "1.0.1",
+               "bindbc-lua": "0.5.0",
+               "jcli": "0.24.0",
+               "lumars": "1.5.0",
                "taggedalgebraic": "0.11.22"
        }
 }
[mydev@fedora lumarsh]$
```

```
[mydev@fedora lumarsh]$ dub build -b release
Performing "release" build using /opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 for aarch64, arm_hardfloat.
jcli:core 0.24.0: target for configuration "library" is up to date.
jcli:introspect 0.24.0: target for configuration "library" is up to date.
jcli:argbinder 0.24.0: target for configuration "library" is up to date.
jcli:argparser 0.24.0: target for configuration "library" is up to date.
jcli:autocomplete 0.24.0: target for configuration "library" is up to date.
jcli:commandparser 0.24.0: target for configuration "library" is up to date.
jcli:text 0.24.0: target for configuration "library" is up to date.
jcli:helptext 0.24.0: target for configuration "library" is up to date.
jcli:resolver 0.24.0: target for configuration "library" is up to date.
jcli 0.24.0: target for configuration "library" is up to date.
bindbc-lua 0.5.0: target for configuration "static" is up to date.
taggedalgebraic 0.11.22: target for configuration "library" is up to date.
lumars 1.5.0+commit.1.ga12c005: building configuration "lua51"...
lumarsh 0.2.4+commit.2.g521a233: building configuration "application"...
Linking...
To force a rebuild of up-to-date targets, run again with --force.
[mydev@fedora lumarsh]$
```

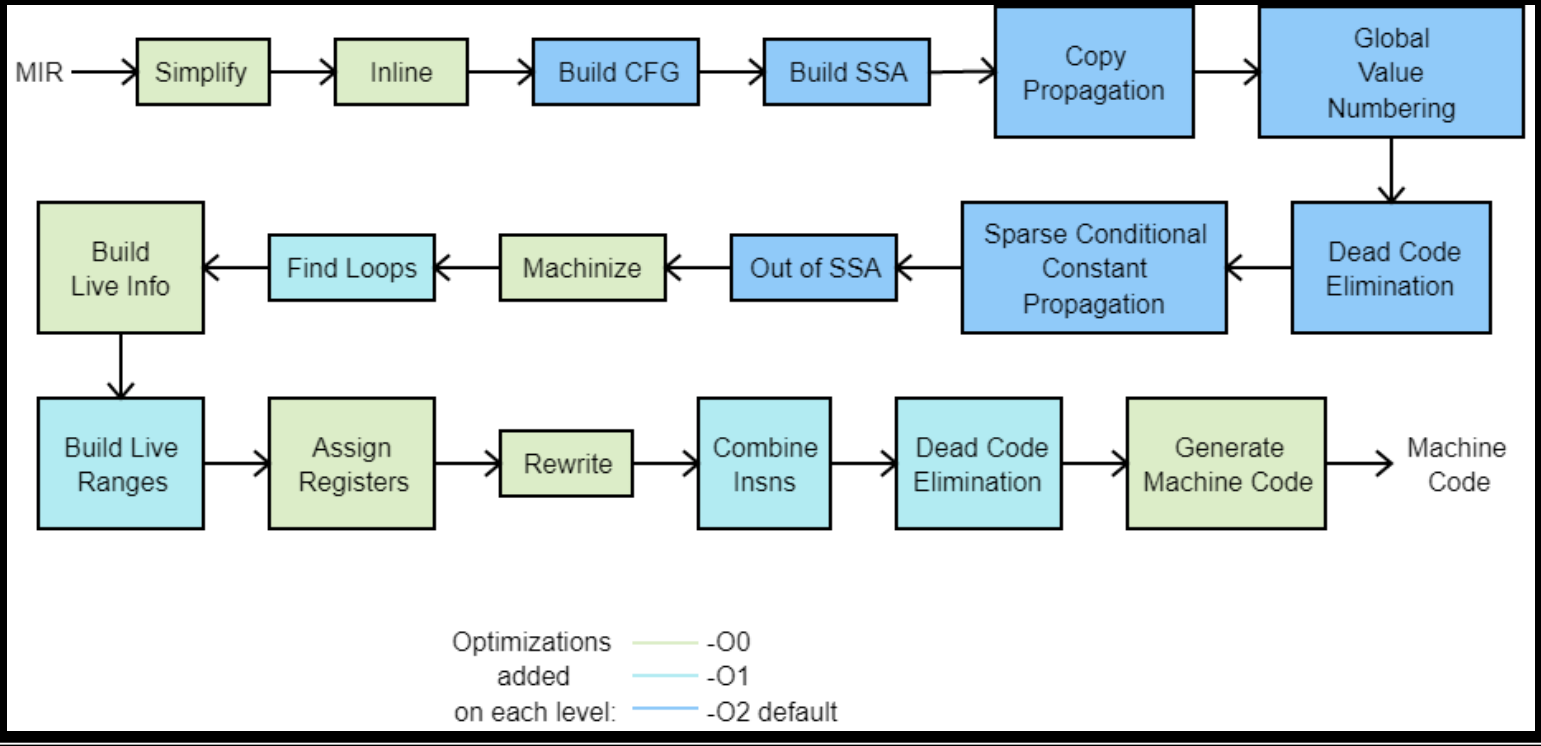
take a glance at the built packages at ~/.dub/packages are:

```
[mydev@fedora /]$ ll ~/.dub/packages
total 0
drwxr-xr-x. 1 mydev mydev 592 Sep  4 10:42 ./
drwxr-xr-x. 1 mydev mydev  22 Aug 27 10:09 ../
drwxr-xr-x. 1 mydev mydev  26 Aug 28 06:26 asdf-0.7.5/
drwxr-xr-x. 1 mydev mydev  62 Aug 27 12:27 bindbc-loader-0.3.2/
drwxr-xr-x. 1 mydev mydev  62 Sep  4 10:10 bindbc-loader-1.0.1/
drwxr-xr-x. 1 mydev mydev  50 Aug 27 12:11 bindbc-lua-0.4.1/
drwxr-xr-x. 1 mydev mydev  50 Sep  4 10:10 bindbc-lua-0.5.0/
drwxr-xr-x. 1 mydev mydev  30 Aug 27 09:55 bolts-1.8.0/
drwxr-xr-x. 1 mydev mydev  30 Aug 27 11:14 bolts-1.8.1/
drwxr-xr-x. 1 mydev mydev  26 Aug 27 12:27 jcli-0.22.4/
drwxr-xr-x. 1 mydev mydev  26 Sep  4 10:42 jcli-0.24.0/
drwxr-xr-x. 1 mydev mydev  34 Aug 27 12:27 lumars-1.1.6/
drwxr-xr-x. 1 mydev mydev  34 Sep  5 03:07 lumars-1.5.0/
drwxr-xr-x. 1 mydev mydev  38 Aug 27 12:11 lumarsh-0.2.4/
drwxr-xr-x. 1 mydev mydev  62 Aug 28 06:26 mir-algorithm-3.10.33/
drwxr-xr-x. 1 mydev mydev  42 Aug 28 06:26 mir-core-1.1.62/
drwxr-xr-x. 1 mydev mydev  54 Aug 26 23:02 openmethods-1.1.0/
drwxr-xr-x. 1 mydev mydev  54 Aug 27 10:51 openmethods-1.3.3/
drwxr-xr-x. 1 mydev mydev  70 Aug 27 12:27 taggedalgebraic-0.11.22/
drwxr-xr-x. 1 mydev mydev  62 Aug 28 06:26 unit-threaded-1.0.11/
drwxr-xr-x. 1 mydev mydev  34 Aug 28 06:26 zeromq-4.2.2/
drwxr-xr-x. 1 mydev mydev  26 Aug 28 06:26 zmqd-1.1.2/
[mydev@fedora /]$
```

- We are still working on idjit, especially for hacking its build system to make it work on RPi4

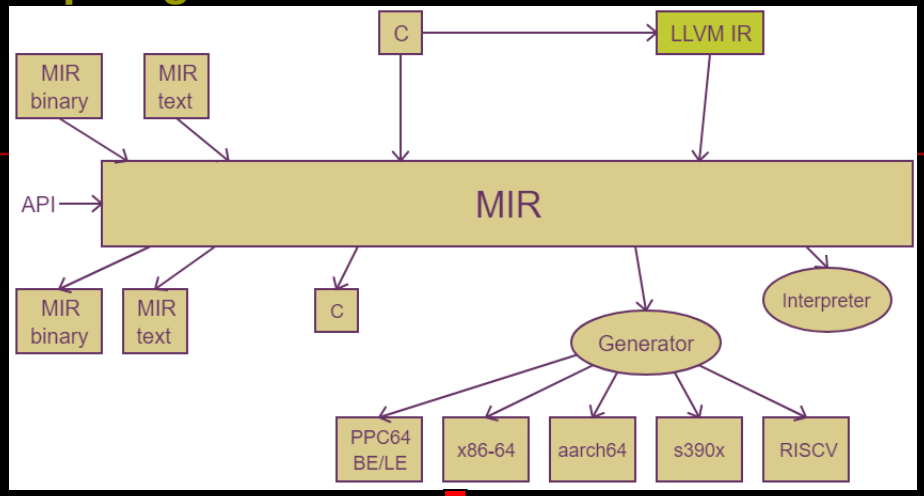
2.2 MIR

- https://github.com/vnmakarov/mir
  A lightweight JIT compiler based on MIR (Medium Internal Representation) and C11 JIT compiler and interpreter based on MIR.
- Motivation
  https://developers.redhat.com/blog/2020/01/20/mir-a-lightweight-jit-compiler-project
- Full JIT compiler pipeline

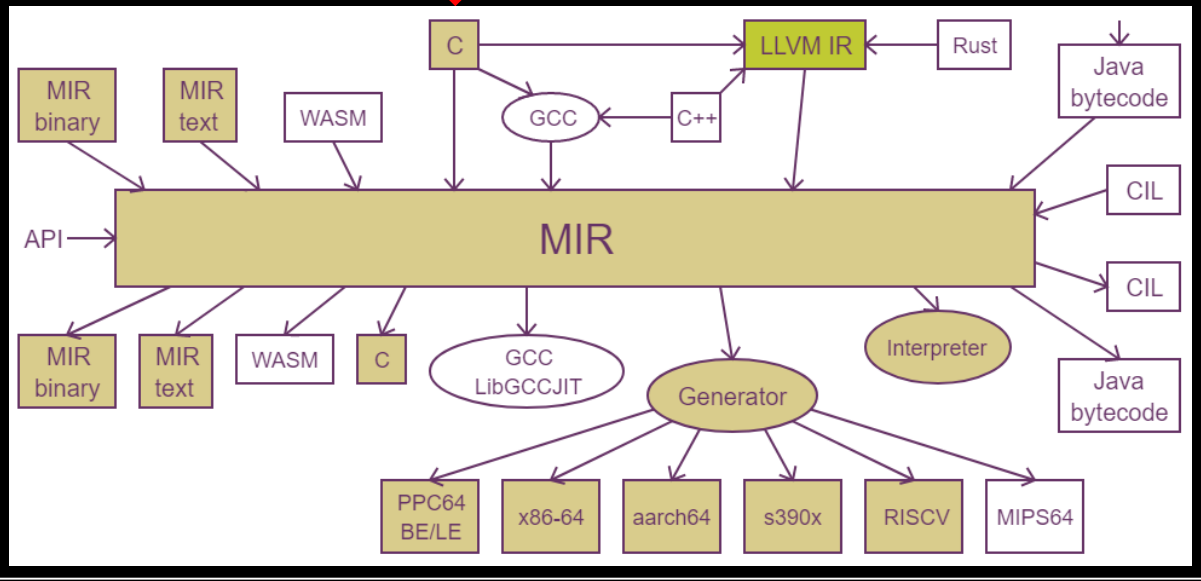


Status

- https://github.com/vnmakarov/mir



current state



possible future state

build MIR on RPi4

```
[mydev@fedora mir]$ make all
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c mir.c -o mir.o
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c mir-gen.c -o mir-gen.o
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c c2mir/c2mir.c -o c2mir/c2mir.o
ar rcs libmir.a mir.o mir-gen.o c2mir/c2mir.o
gcc -shared -Wl,-soname,libmir.so.0 -o libmir.so.0.0.3 mir.o mir-gen.o c2mir/c2mir.o
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c c2mir/c2mir-driver.c -o c2mir/c2mir-driver.o
gcc  c2mir/c2mir.o c2mir/c2mir-driver.o libmir.a -lm -ldl -lpthread -o c2m
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c mir-utils/m2b.c -o mir-utils/m2b.o
gcc  mir-utils/m2b.o libmir.a -lm -ldl -lpthread -o m2b ./libmir.a
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c mir-utils/b2m.c -o mir-utils/b2m.o
gcc  mir-utils/b2m.o libmir.a -lm -ldl -lpthread -o b2m ./libmir.a
gcc -I. -MMD -MP -fPIC -g -std=gnu11 -Wno-abi -fsigned-char -fno-tree-sra -fno-ipa-cp-clone -O3 -DNDEBUG -DMIR_PARALLEL_GEN
-c mir-utils/b2ctab.c -o mir-utils/b2ctab.o
gcc  mir-utils/b2ctab.o libmir.a -lm -ldl -lpthread -o b2ctab  ./libmir.a
[mydev@fedora mir]$
```

```
[mydev@fedora mir]$ tree ./build/
./build/
├── bin
│   ├── b2ctab
│   ├── b2m
│   ├── c2m
│   └── m2b
├── include
│   ├── c2mir.h
│   ├── mir-dlist.h
│   ├── mir-gen.h
│   ├── mir.h
│   ├── mir-htab.h
│   └── mir-varr.h
└── lib
    ├── libmir.a
    ├── libmir.so.0 → /opt/MyWorkSpace/MyProjs/Runtime/JIT/MIR/mir/build/lib/libmir.so.0.0.3
    └── libmir.so.0.0.3
```

3) How about DLLVM

This is a long-term plan, and here are some initial ideas:

- A lightweight version of LLVM in  .

- A re-implementation of LLVM in with limited platforms, features, targets and tools support , while that for Linux, MLIR, ARM/RISCV/ Wasm/eBPF/SPIR-V has top priority.

- Re-implement Clang with a customized DMD-FE, and only support D, C, OpenCL C.

- …

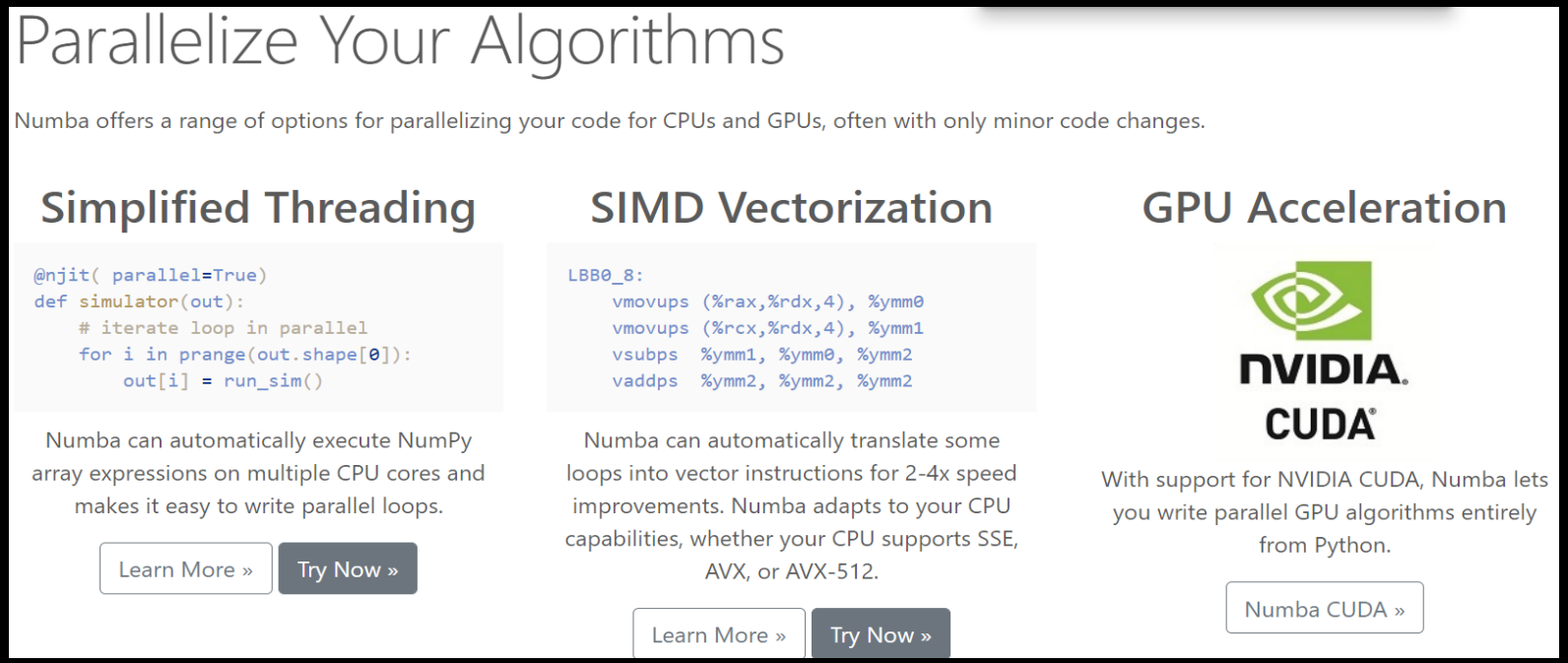
Considering a combination of the IR design from Mir and Gyre…

Let's enhance LDC to better support for ARM, RISC-V, Wasm, eBPF and SPIR-V firstly!

IV. Parallel Computing with D

Reference Projects

- **https://numba.pydata.org/**
  https://github.com/numba/numba
  NumPy aware dynamic Python compiler using LLVM.



  https://github.com/numba/llvmlite
- **https://github.com/taichi-dev/taichi**
- **…**

1) Libmir

- https://github.com/libmir

Generic Numerical Library for Science and Machine Learning.

Mir Core - Base software building blocks: Algebraic types, universal reflection API, basic math, and more.

Numir - NumPy-like API wrappers of Mir.

Separated Mir Projects

- Mir Algorithm - Multidimensional arrays (ndslice), iterators, algorithms.
- Mir Random - Professional Random Number Generators
- Mir GLAS - Linear Algebra Library (Experimental, not supported for now)
- Mir BLAS - Bindings to libraries with CBLAS API like OpenBLAS and Intel MKL.
- Mir LAPACK - Bindings to libraries with LAPACK API like OpenBLAS and Intel MKL.
- Mir Optim - Nonlinear Solvers.
- Mir CPUID - CPU Identification routines (less buggy then Phobos).

Mir Ion - Dlang Serialization Framework.

Mir Stat - Dlang Statistical Package.

DCV - Computer Vision Library for D Programming Language.

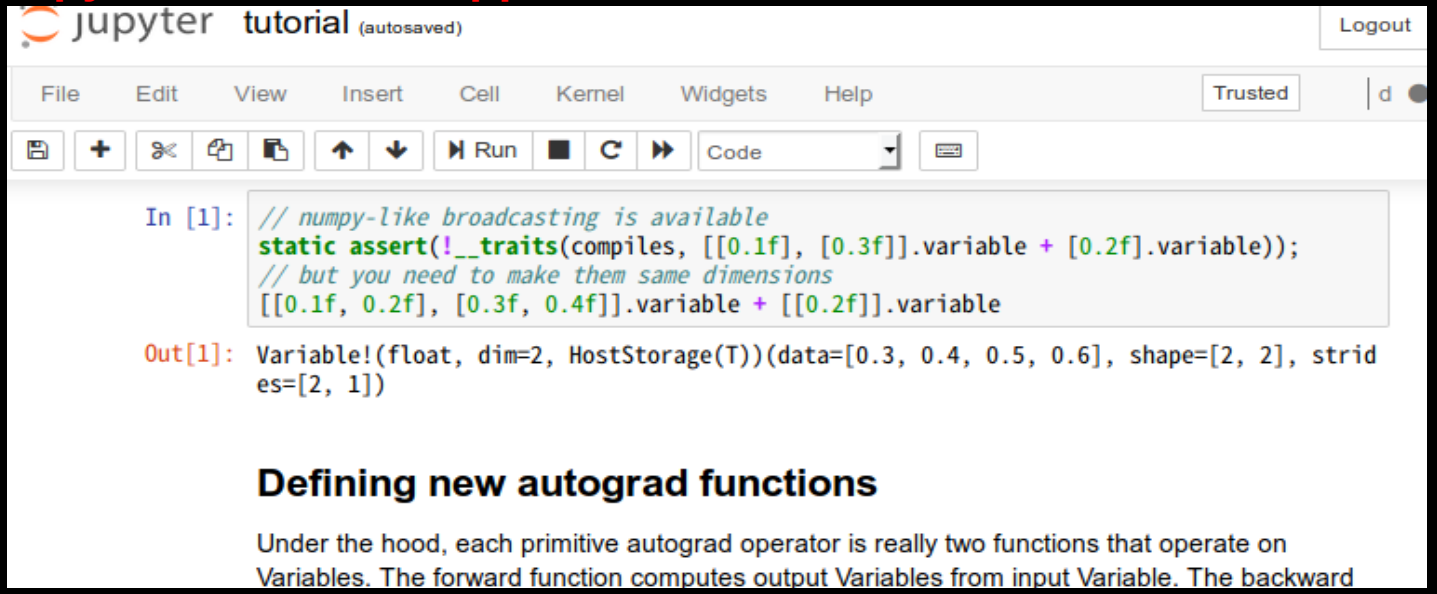
…

Grain

- https://github.com/ShigekiKarita/grain
  Autograd mir and CUDA library for dynamic neural networks in D.
- Features

- dynamic computation graph like chainer or pytorch
- statically typed tensor `Variable(T, size_t dim, alias Storage)` unlike numpy
- CPU (mir) and CUDA (cublas/cudnn) backend
- extensible (i.e., user-defined) autograd function
- LDC2 (CPU/CUDA) and DMD (CPU only) support

- Jupyter notebook support



Source: https://github.com/ShigekiKarita/grain-talk/blob/master/slide.pdf

Try to build jupyterd on RPi4

```
[mydev@fedora jupyterd]$ ./install.sh
Fetching drepl 0.2.1 (getting selected version)...
Fetching colorize 1.0.5 (getting selected version)...
Fetching stdx-allocator 2.77.5 (getting selected version)...
Fetching linenoise 1.1.0+1.0.0 (getting selected version)...
Fetching asdf 0.2.5 (getting selected version)...
Failed to download package asdf from https://code.dlang.org/packages/asdf/0.2.5.zip
cp: cannot stat './jupyterd': No such file or directory
[mydev@fedora jupyterd]$
```

```
[mydev@fedora jupyterd]$ ./install.sh
Fetching asdf 0.2.5 (getting selected version)...
Fetching libdparse 0.8.8 (getting selected version)...
Performing "debug" build using /opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 for aarch64, arm_hardfloat.
asdf 0.2.5: building configuration "library"...
'-sse4.2' is not a recognized feature for this target (ignoring feature)
Info: SSE4.2 instructions are not used for ASDF.
/home/mydev/.dub/packages/asdf-0.2.5/asdf/source/asdf/outputarray.d(84,4): Error: static assert:  "not implemented for this
target"
/opt/MyWorkSpace/DevSW/D/LDC/1.30.0/bin/ldc2 failed with exit code 1.
cp: cannot stat './jupyterd': No such file or directory
[mydev@fedora jupyterd]$
```

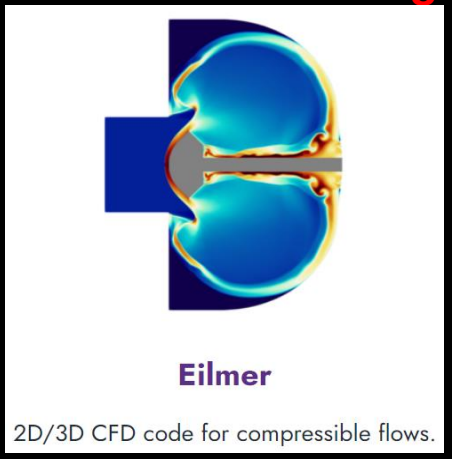
the build error shows that jupyterd only support X86…

2) GDTk

- https://gdtk.uqcloud.net/

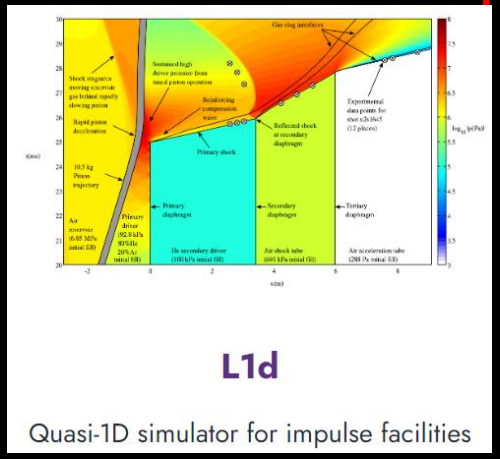
  Gas Dynamics Toolkit

  A collection of software for doing gas dynamics, from simple desktop calculations through to simulations on supercomputers



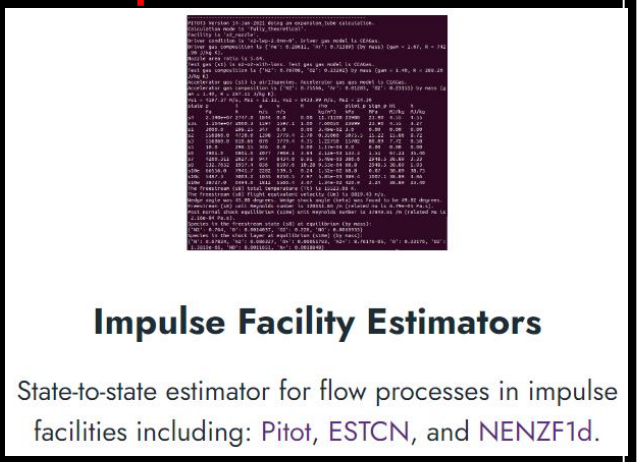
Eilmer

2D/3D CFD code for compressible flows.



L1d

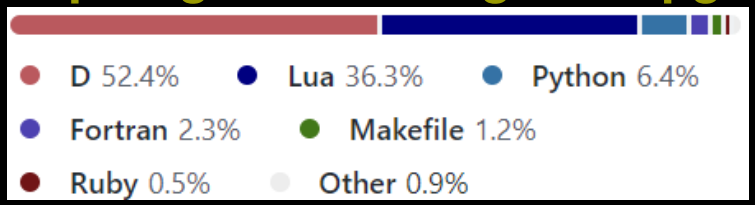
Quasi-1D simulator for impulse facilities



Impulse Facility Estimators

State-to-state estimator for flow processes in impulse facilities including: Pitot, ESTCN, and NENZF1d.

- https://github.com/gdtk-uq/gdtk



- https://zhuanlan.zhihu.com/p/50486526 //Eilmer— 可压缩流体仿真工具

Rewritten in D

- **https://dlang.org/blog/2022/02/02/a-gas-dynamics-toolkit-in-d/**

Some history

This simulation program, originally called cns4u, started as a relatively small C program that ran on the Cray-Y/MP supercomputer at NASA Langley Research Center in 1991. A PDF of an early report with the title 'Single-Block Navier-Stokes Integrator' can be found here. It describes the simple finite-volume formulation of the code that allows simulation of a nonreacting gas on a single, structured grid. Thirty years on, many capabilities have been added through the efforts of a number of academic staff and students. These capabilities include high-temperature thermochemical effects with reacting gases and distributed-memory parallel simulations on cluster computers. The language in which the program was written changed from C to C++, with connections to Tcl, Python and Lua.

The motivation for using C++ in combination with the scripting languages was to allow many code variations and user programmability so that we could tackle any number of initially unimagined gas-dynamic processes as new PhD students arrived to do their studies. By 2010, the Eilmer3 code (as it was called by then) was sitting at about 100k lines of code and was growing. We were, and still are, mechanical engineers and students of gas-dynamics first and programmers second. C++ was a lot of trouble for us. Over the next 4 years, C++ became even more trouble for us as the Eilmer3 code grew to about 250k lines of code and many PhD students used it to do all manner of simulations for their thesis studies.

Also in 2010, a couple of us (PAJ and RJG) living in different parts of the world (Queensland, Australia and Virginia, USA) came across the D programming language and took note of Andrei Alexandrescu's promise of stability into the future. Here was the promise of a C++ replacement that we could use to rebuild our code and remain somewhat sane. We each bought a copy of Andrei's book and experimented with the D language to see if it really was the *C++-done-right* that we wished for. One of us still has the copy of the initial printing of Andrei's book without his name on the front cover.

Rebuilding in D

In 2014 we got serious about using D for the next iteration of Eilmer and started porting the core gas dynamics code from C++ to D. Over the next four years, in between university teaching activities, we reimplemented much of the Eilmer3 C++ code in D and extended it. We think that this was done to good effect. This conference paper, from late 2015, documents our effort at the initial port of the structured grid solver. (A preprint is hosted on our site.) The Eilmer4 program is as fast as the earlier C++ program but is far more versatile while being implemented in fewer lines of code. It now works with unstructured as well as structured grids and has a new flexible boundary condition model, a high-temperature thermochemistry module, and in the past two years we have added the Newton-Krylov-accelerated steady-state solver that was used to do the simulation shown above. And importantly for us, with the code now being in D, we now have have many fewer *WTF* moments.

Features of D that have been of benefit to us include:

- Template programming that other Mechanical Engineers can understand (thanks Walter!). Many of our numerical routines are defined to work with `numbers` that we define as an alias to either `double` or `Complex!double` values. This has been important to us because we can use the same basic update code and get the sensitivity coefficients via finite differences in the complex direction. We think this saved us a large number of lines of code.

- String mixins have replaced our use of the M4 preprocessor to generate C++ code in Eilmer3. We still have to do a bit of head-scratching while building the code with mixins, but we have retained most of our hair—something that we did not expect to do if we continued to work with C++.

- Good error messages from the compiler. We often used to be overwhelmed by the C++ template error messages that could run to hundreds of lines. The D compilers have been much nicer to us and we have found the “did you mean” suggestions to be quite useful.

- A comprehensive standard library in combination with language features such as delegates and closures that allow us to write less code and instead concentrate on our gas dynamics calculations. I think that having to use C++ Functors was about the tipping point in our 25-year adventure with C++.

- Ranges and the foreach loops make our D code so much tidier than our equivalent C++ code.

- Low-barrier shared-memory parallelism. We do many of the flow update calculations in parallel over blocks of cells and we like to take advantage of the many cores that are available on a typical workstation.

- Simple and direct linkage to C libraries. We make extensive use of Lua for our configuration and do large simulations by using many processors in parallel via the OpenMPI library.

- The garbage collector is wonderful, even if other people complain about it. It makes life simpler for us. For the input and output, we take the comfortable path of letting the compiler manage the memory and then tell the garbage collector to tidy up after us. Of course, we don't want to overuse it. @nogc is used in the core of the code to force us not to generate garbage. We allocate much of our data storage at the start of a simulation and then pass references to parts of it into the core functions.

- Fast compilation and good optimizing compilers. Nearly an hour's build time was fairly common for our old C++ code, and now we would expect a DMD or LDC debug build in about a quarter of a minute. This builds a basic version of the main simulation code on a Lenovo ThinkPad with Core i7 processor. An optimized build can take a little over a minute but the benefit of the faster simulation is paid back by orders of magnitude when a simulation job is run for several hours over hundreds of processors.

- `version(xxxx) { ... }` has been a good way to have variants of the code. Some use complex numbers and others are just double numbers. Also, some variants of the code can have multiple chemical species and/or just work with a single-species nonreacting gas. This reduction in physical modelling allows us to reduce the memory required by the simulation. For big simulations of 3D flows, the required memory can be on the order of hundreds of gigabytes.

- `debug { ... }` gets used to hide IO code in `@nogc` functions. If a simulation fails, our first action is often to run the debug-flavor of the code to get more information and then, if needed, run the debug flavor of the code under the control of `gdb` to dig into the details.

3) DCompute

- https://github.com/libmir/dcompute

DCompute: Native execution of D on GPUs and other Accelerators.

This project is a set of libraries designed to work with LDC to enable native execution of D on GPUs (and other more exotic targets of OpenCL such as FPGAs DSPs, hereafter just 'GPUs') on the OpenCL and CUDA runtimes. As DCompute depends on developments in LDC for the code generation, a relatively recent LDC is required, use 1.8.0 or newer.

There are four main parts:

- std: A library containing standard functionality for targetting GPUs and abstractions over the intrinsics of OpenCL and CUDA.
- driver: For handling all the compute API interactions and provide a friendly, easy-to-use, consistent interface. Of course you can always get down to a lower level of interaction if you need to. You can also use this to execute non-D kernels (e.g. OpenCL or CUDA).
- kernels: A set of standard kernels and primitives to cover a large number of use cases and serve as documentation on how (and how not) to use this library.
- tests: A framework for testing kernels. The suite is runnable with `dub test` (see `dub.json` for the configuration used).

An example

Kernel:

```
@kernel void saxpy(GlobalPointer!(float) res,
                   float alpha,
                   GlobalPointer!(float) x,
                   GlobalPointer!(float) y,
                   size_t N)
{
    auto i = GlobalIndex.x;
    if (i >= N) return;
    res[i] = alpha*x[i] + y[i];
}
```

Invoke with (CUDA):

```
q.enqueue!(saxpy)
    ([N,1,1],[1,1,1]) // Grid & block & optional shared memory
    (b_res,alpha,b_x,b_y, N); // kernel arguments
```

equivalent to the CUDA code

```
saxpy<<<1,N,0,q>>>(b_res,alpha,b_x,b_y, N);
```

For more examples and the full code see `source/dcompute/tests`.

4) Rethinking

- Project Mir has a good base for further improvements.
- Evaluating extend DCompute to support more hardware like FPGA and NPU in the future.

Try to support new DSL for kernels development via self-defined annotations, e.g.:
@kernel.xx.dsl or @kernel(dsl=yy)
and more.

- ROCm + HIP(AMD) has great potential when compared with the other heterogeneous parallel computing framework like CUDA(Nivdia) and OneAPI(Intel) for openness and design.
- Let's go on with project jupyter-wire
(https://github.com/symmetryinvestments/jupyter-wire)
for an enhanced D-based Jupyter kernel implementation.

- # Rewrite Z3(Theorem Prover) in D?

https://github.com/Z3Prover/z3

Stats (master branch with last commit: dd91fab6f434aff5aef0db7886a23f8ddc90c261)

```
[mydev@fedora z3]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 Autoconf               16         2880          560         1707          613
 Batch                   2          106           86            0           20
 C                       3         8113         5583         1262         1268
 C Header              981       188993       122491        41157        25345
 CMake                  99         5318         4090          912          316
 C#                     96        22645        12091         7731         2823
 C++                   892       448168       363663        40300        44205
 D                       1          439          365           38           36
 Dockerfile              1           45           24           17            4
 Java                   92        18715         9549         6913         2253
 JavaScript              2           46           37            4            5
 JSON                    8        11476        11476            0            0
 Makefile                1            7            5            0            2
 MSBuild                 6          388          354           25            9
 OCaml                   3         5929         3004         1737         1188
 Python                 49        26377        22344          987         3046
 Shell                   4           60           39           11           10
 Plain Text             10          498            0          393          105
 TOML                    1            3            3            0            0
 TypeScript             18         5549         4144          805          600
 Visual Studio Pro|      1          137          137            0            0
 Visual Studio Sol|      1          125          124            0            1
 YAML                   13         1691         1543           73           75
-------------------------------------------------------------------------------
 HTML                    5         2965         2544            0          421
 ├─ JavaScript           1            1            1            0            0
 (Total)                              2966         2545            0          421
-------------------------------------------------------------------------------
 Jupyter Notebooks       3            0            0            0            0
 ├─ Markdown             1          176            0          115           61
 ├─ Python               1          529          384           60           85
 (Total)                              705          384          175          146
-------------------------------------------------------------------------------
 Markdown               12         3007            0         2286          721
 ├─ BASH                 1           28           27            1            0
 ├─ C                    1            2            2            0            0
 ├─ JavaScript           1           15           12            1            2
 ├─ TOML                 1            2            2            0            0
 ├─ TypeScript           1           12            8            2            2
 (Total)                              3066           51         2290          725
===============================================================================
 Total                2320       753680       564256       106358        83066
===============================================================================
[mydev@fedora z3]$
```

V. D for HW-SW Co-design/FOSS EDA

1) D-based DSL

1.1 Vox

- https://github.com/MrSmith33/vox

  Vox language compiler. AOT/JIT/Linker. Zero dependencies. A multiparadigm programming language inspired by D(60%), Jai(30%), and Zig(10%).

- Languages

- Main features

  - Fast compilation
  - Strong metaprogramming
  - Can be used for scripting and standalone programs (both JIT and AOT compilation)
  - No dependencies (except D compiler)

Platforms

Supported:

- windows-x64 - host and target
- linux-x64 - host and target
- macos-x64 - only jit-mode

Planned:

- linux-arm64
- wasm
- windows-arm64?
- spirv (Vulkan/OpenCL/OpenGL shaders)

Differences from 

Similarities to the D language

- Same syntax for most portions of the language (struct, function, enum, for, while, if, UFCS, slices, arrays)
- Conditional compilation
- Templates, Variadic templates, and functions
- C interoperability
- Modules / Packages

Differences from the D language

- No GC, minimal runtime, no classes (only structs), no exceptions
- More compile-time features, faster CTFE
- Using templates for heavy calculations is discouraged, instead, CTFE can be used for introspection, and code generation.
- Macros (WIP)
- No C++ interoperability

Design

Project goals

- Strong focus on application extensions
- Maximize user productivity
- Maximize application performance
- AOT and JIT, plugin support, runtime compilation, embedded compiler, tiered compilation
- Static typing
- Great error messages
- Fast / Incremental compilation
- Minimize effort needed for installation, setup, integration
  - Minimal dependencies/encapsulate dependency into module or package
  - Runtime as a library/minimal runtime/no runtime
  - Embedding/extern C
  - Code driven compilation, extending compiler from inside of the program being compiled
- Processor intrinsics
- Conditional compilation
- CTFE, Templates, Introspection, Code generation

https://github.com/MrSmith33/vox/blob/master/internals.md

Examples

```
i32 fib(i32 number) {
    if (number < 1) return 0;
    if (number < 3) return 1;
    return fib(number-1) + fib(number-2);
}

struct Point {
    i32 x;
    i32 y;
}

T min[T](T a, T b) {
    if (a < b) return a;
    return b;
}
```

- Example of JIT compilation for amd64 from D code:

▾ code

```
// Source code
string source = q{
    void test(i32* array, i32 index, i32 value) {
        array[index] = value;
    }
};

// Error handling is omitted
Driver driver;
driver.initialize(jitPasses);
scope(exit) driver.releaseMemory;
driver.beginCompilation();
driver.addModule(SourceFileInfo("test", source));
driver.compile();
driver.markCodeAsExecutable();

// Get function pointer
auto testFun = driver.context.getFunctionPtr!(void, int*, int, int)("test");

// Use compiled function
int[2] val = [42, 56];
testFun(val.ptr, 1, 10);
assert(val[1] == 10);
```

- Voxel engine that uses Vox as a scripting language: Voxelman 2

https://github.com/MrSmith33/voxelman2

2) System Modeling

Overview

- A perspective from Vlang

System Modeling Prowess

C++ function/operator overloading and generic programming paradigm make it easy for coders to define new data types in the language. SystemVerilog, on the other hand provides first class dynamic arrays, associative arrays and implements garbage collection. These SystemVerilog features allow users to code at higher abstraction levels and to focus on the domain logic. D Programming Language brings all these strengths under one hood. E-UVM inherits these strengths from D and adds hardware modeling capabilities.

- First class dynamic arrays and associative array data types
- Best in the world generic and generative programming capabilities
- E-UVM offers capability to model multiple asynchronous systems by letting the user create multiple simulator systems running in parallel

Source: http://uvm.io/

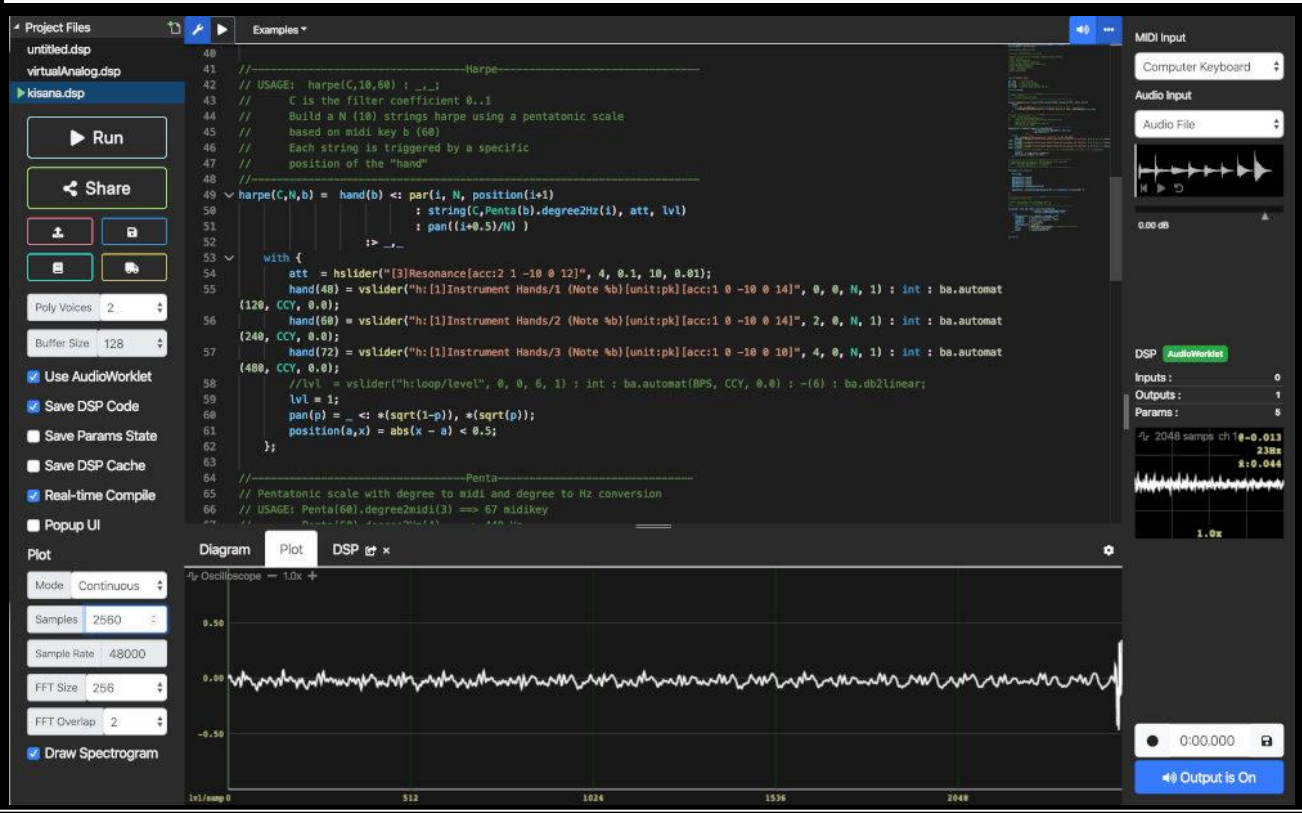
2.1 Faust

- https://faust.grame.fr/

Programming Language for Audio Applications and Plugins.

Faust (Functional Audio Stream) is a functional programming language for sound synthesis and audio processing with a strong focus on the design of synthesizers, musical instruments, audio effects, etc. Faust targets high-performance signal processing applications and audio plug-ins for a variety of platforms and standards.

The core component of Faust is its compiler. It allows to "translate" any Faust digital signal processing (DSP) specification to a wide range of non-domain specific languages such as C++, C, LLVM bit code, WebAssembly, Rust, etc. In this regard, Faust can be seen as an alternative to C++ but is much simpler and intuitive to learn.



Design Principles

- FAUST is a *specification language*. It aims at providing an adequate notation to describe *signal processors* from a mathematical point of view. FAUST is, as much as possible, free from implementation details.

- FAUST programs are fully compiled, not interpreted. The compiler translates FAUST programs into equivalent C++ programs taking care of generating the most efficient code. The result can generally compete with, and sometimes even outperform, C++ code written by seasoned programmers.

- The generated code works at the sample level. It is therefore suited to implement low-level DSP functions like recursive filters. Moreover the code can be easily embedded. It is self-contained and doesn't depend of any DSP library or runtime system. It has a very deterministic behavior and a constant memory footprint.

- The semantic of FAUST is simple and well defined. This is not just of academic interest. It allows the FAUST compiler to be *semantically driven*. Instead of compiling a program literally, it compiles the mathematical function it denotes. This feature is useful for example to promote components reuse while preserving optimal performance.

- FAUST is a textual language but nevertheless block-diagram oriented. It actually combines two approaches: *functional programming* and *algebraic block-diagrams*. The key idea is to view block-diagram construction as function composition. For that purpose, FAUST relies on a *block-diagram algebra* of five composition operations (`: , ~ <: :>`).

- Thanks to the notion of *architecture*, FAUST programs can be easily deployed on a large variety of audio platforms and plugin formats without any change to the FAUST code.

Source: https://github.com/grame-cncm/faust/blob/master-dev/documentation/faust-quick-reference.pdf

Dlang in Faust

```
[mydev@fedora faust]$ find ./architecture -name "*.d"
./architecture/bench.d
./architecture/dplug.d
./architecture/minimal.d
./architecture/params.d
[mydev@fedora faust]$
```

```
// faust -a dplug.d -lang dlang noise.dsp -o noise.d
```

```
[mydev@fedora faust]$ tree -d compiler/generator
compiler/generator
├── c
├── cpp
├── csharp
├── dlang
├── fir
├── interpreter
├── java
├── julia
├── llvm
├── rust
├── soul
└── wasm
```

...

3) D for Virtual Platform

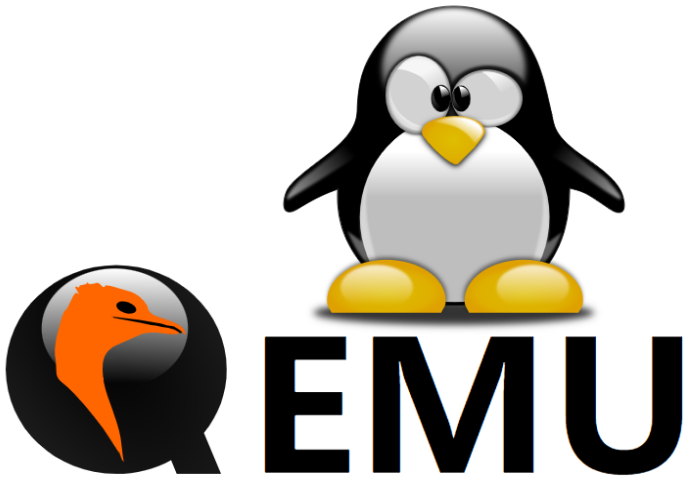
A perspective from Vlang

- ## Emulation and Virtual Platform

SystemVerilog was designed with sole focus on RTL verification. It integrates tightly and efficiently with RTL. On the other hand Emulation and Virtual platforms generally have a C level API. Due to DPI overhead and inherent slowness, SystemVerilog becomes a bottleneck when used for testbenching these platforms.

- E-UVM can potentially generate transactions at multiple orders of magnitude faster compared to SystemVerilog -- thanks to native compilation and multicore abilities
- E-UVM offers zero integration overhead when testbenching virtual and emulation platforms -- thanks to its underlying C compatible ABI

Take a look at E-UVM/Qemu interface demo ► | Back to Top ►



D for Wasm

- https://wiki.dlang.org/Generating_WebAssembly_with_LDC

Let's generate a .wasm file for this D code (`wasm.d`):

```
extern(C): // disable D mangling

double add(double a, double b) { return a + b; }

// seems to be the required entry point
void _start() {}
```

Build `wasm.wasm`:

```
ldc2 -mtriple=wasm32-unknown-unknown-wasm -betterC wasm.d
```

If using DUB:

```
dub build --compiler=ldc2 --arch=wasm32-unknown-unknown-wasm
```

In case LDC errors out (e.g., with unsupported `-link-internally`), try an official prebuilt release package.

Additional LLVM wasm features

You can list the supported LLVM features for WASM with your LDC version like SIMD or exception handling using

```
ldc2 -mtriple=wasm32-unknown-unknown-wasm -mattr=help
```

Example output: (**may be outdated**, run the above command yourself to get the supported features for your LDC version)

```
Targeting wasm32. Available CPUs for this target:

  bleeding-edge - Select the bleeding-edge processor.
  generic       - Select the generic processor.
  mvp           - Select the mvp processor.

Available features for this target:

  atomics              - Enable Atomics.
  bulk-memory          - Enable bulk memory operations.
  exception-handling   - Enable Wasm exception handling.
  multivalue           - Enable multivalue blocks, instructions, and functions.
  mutable-globals      - Enable mutable globals.
  nontrapping-fptoint  - Enable non-trapping float-to-int conversion operators.
  reference-types      - Enable reference types.
  sign-ext             - Enable sign extension operators.
  simd128              - Enable 128-bit SIMD.
  tail-call            - Enable tail call instructions.

Use +feature to enable a feature, or -feature to disable it.
For example, llc -mcpu=mycpu -mattr=+feature1,-feature2
```

- ...

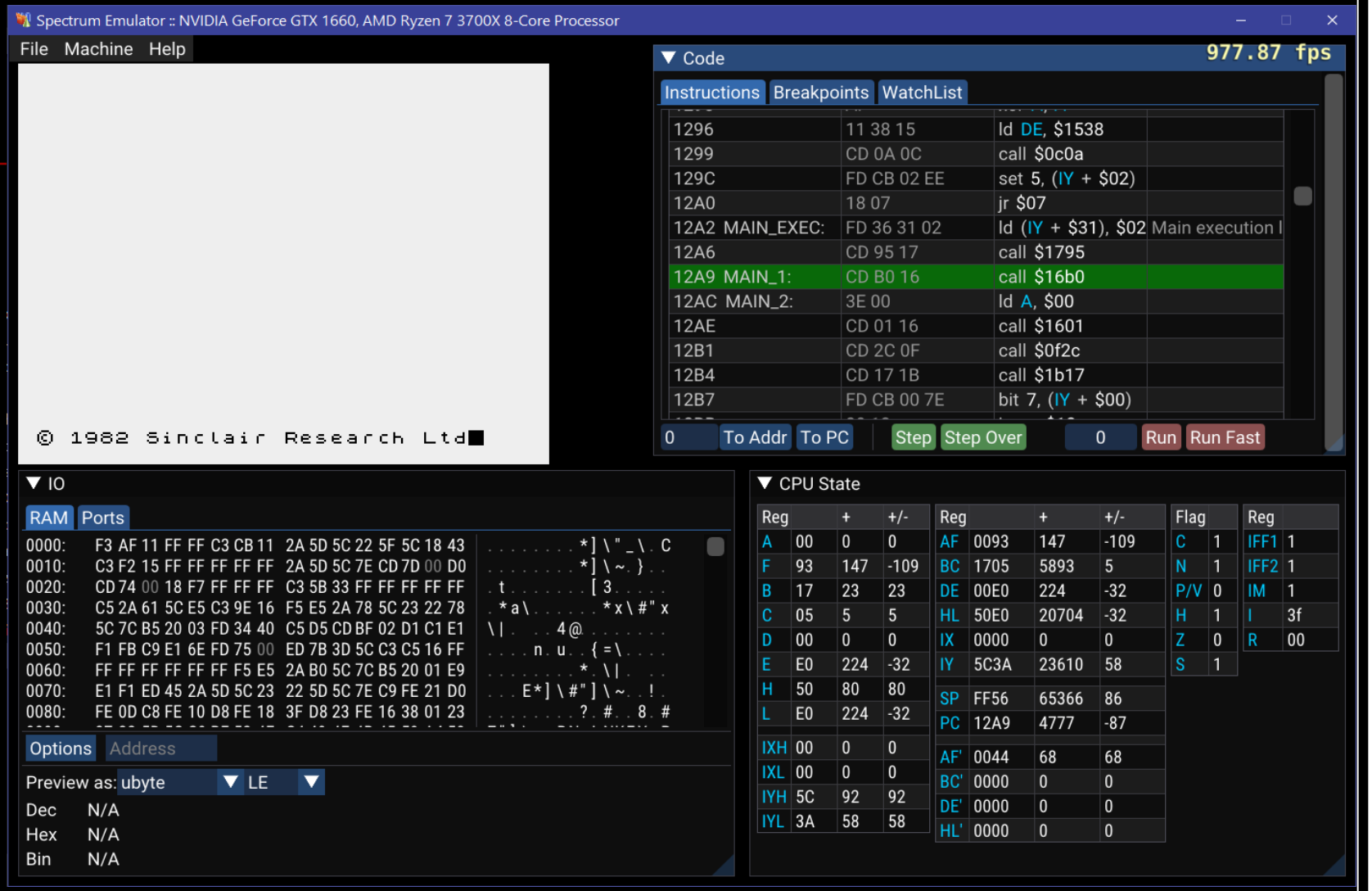
3.1 Emulators

3.1.1 Emulator

- https://github.com/pvmoore/Emulator
  Chip and machine emulation: currently emulates MOS 6502 processor and Zilog Z80 processor.
- Languages

- Dependencies
  - Dlang-common https://github.com/pvmoore/dlang-common
  - Dlang-logging https://github.com/pvmoore/dlang-logging
  - Dlang-events https://github.com/pvmoore/dlang-events
  - Dlang-vulkan https://github.com/pvmoore/dlang-vulkan
  - CImgui DLL
  - GLFW3 DLL
  - Vulkan DLL

Screenshot

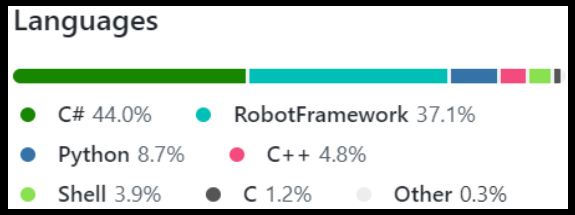

Spectrum Emulator :: NVIDIA GeForce GTX 1660, AMD Ryzen 7 3700X 8-Core Processor
File Machine Help
© 1982 Sinclair Research Ltd
Code
977.87 fps
Instructions Breakpoints WatchList
1296 11 38 15 ld DE, $1538
1299 CD 0A 0C call $0c0a
129C FD CB 02 EE set 5, (IY + $02)
12A0 18 07 jr $07
12A2 MAIN_EXEC: FD 36 31 02 ld (IY + $31), $02 Main execution l
12A6 CD 95 17 call $1795
12A9 MAIN_1: CD B0 16 call $16b0
12AC MAIN_2: 3E 00 ld A, $00
12AE CD 01 16 call $1601
12B1 CD 2C 0F call $0f2c
12B4 CD 17 1B call $1b17
12B7 FD CB 00 7E bit 7, (IY + $00)
0 To Addr To PC Step Step Over 0 Run Run Fast
IO
RAM Ports
0000: F3 AF 11 FF FF C3 CB 11 2A 5D 5C 22 5F 5C 18 43
0010: C3 F2 15 FF FF FF FF FF 2A 5D 5C 7E CD 7D 00 D0
0020: CD 74 00 18 F7 FF FF FF C3 5B 33 FF FF FF FF FF
0030: C5 2A 61 5C E5 C3 9E 16 F5 E5 2A 78 5C 23 22 78
0040: 5C 7C B5 20 03 FD 34 40 C5 D5 CD BF 02 D1 C1 E1
0050: F1 FB C9 E1 6E FD 75 00 ED 7B 3D 5C C3 C5 16 FF
0060: FF FF FF FF FF FF F5 E5 2A B0 5C 7C B5 20 01 E9
0070: E1 F1 ED 45 2A 5D 5C 23 22 5D 5C 7E C9 FE 21 D0
0080: FE 0D C8 FE 10 D8 FE 18 3F D8 23 FE 16 38 01 23
Options Address
Preview as: ubyte LE
Dec N/A
Hex N/A
Bin N/A
CPU State
Reg + +/-
A 00 0 0
F 93 147 -109
B 17 23 23
C 05 5 5
D 00 0 0
E E0 224 -32
H 50 80 80
L E0 224 -32
IXH 00 0 0
IXL 00 0 0
IYH 5C 92 92
IYL 3A 58 58
Reg + +/-
AF 0093 147 -109
BC 1705 5893 5
DE 00E0 224 -32
HL 50E0 20704 -32
IX 0000 0 0
IY 5C3A 23610 58
SP FF56 65366 86
PC 12A9 4777 -87
AF' 0044 68 68
BC' 0000 0 0
DE' 0000 0 0
HL' 0000 0 0
Flag
C 1
N 1
P/V 0
H 1
Z 0
S 1
Reg
IFF1 1
IFF2 1
IM 1
I 3f
R 00

3.2 How about re-implement Renode in D?

3.2.1 Src of Renode

- A virtual development framework for complex embedded systems which is RISC-V friendly.
- https://github.com/renode/renode

Stats without submodules (last commit: d136ec13703f175fdf5eb94aa9b2c851570994a4)




```
[mydev@fedora renode-master-no-submodules]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 BASH                    3           98           74            5           19
 Batch                   2           47           33            4           10
 C                       1          235          162           33           40
 C Header               13          750          536          109          105
 CMake                   1            3            1            1            1
 C#                    104        14251        11673          887         1691
 C++                    11         1803         1271          240          292
 MSBuild                 4          578          577            1            0
 PowerShell              2           18           14            2            2
 Python                 26         2684         2101           79          504
 ReStructuredText        9         1567         1211            0          356
 Shell                  19         1374         1031          113          230
 SVG                     1           42           42            0            0
 Plain Text              3           11            0           11            0
 Visual Studio Sol|      1          582          581            0            1
 YAML                    2          147          140            1            6
-------------------------------------------------------------------------------
 Markdown                3          371            0          256          115
 ├─ INI                  1            8            7            0            1
 ├─ Shell                1           20           20            0            0
 (Total)                            399           27          256          116
===============================================================================
 Total                 205        24561        19447         1742         3372
===============================================================================
[mydev@fedora renode-master-no-submodules]$
```

Stats with submodules (last commit: d136ec13703f175fdf5eb94aa9b2c851570994a4)

```
[mydev@fedora renode-master-with-submodules]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 BASH                    7          270          169           41           60
 Batch                   2           47           33            4           10
 C                      95        96568        80685         7444         8439
 C Header              110        67997        58246         4958         4793
 CMake                   3          184          138            5           41
 C#                   5958       976914       722106       130617       124191
 C++                    11         1803         1271          240          292
 HTML                    3         1089         1058            1           30
 JSON                    1           17           17            0            0
 Makefile                2           22           14            0            8
 MSBuild               120        38580        38447           92           41
 PowerShell              2           18           14            2            2
 Python                 26         2684         2101           79          504
 ReStructuredText       13         1661         1278            0          383
 Shell                  23         1477         1107          124          246
 SVG                     1           42           42            0            0
 Plain Text             36      1560645            0      1557805         2840
 Visual Studio Sol|     22         2459         2438            0           21
 XAML                    4          278          256            0           22
 XML                   441        65859        65857            2            0
 YAML                    4          255          238            1           16
-------------------------------------------------------------------------------
 Markdown               14         1314            0          936          378
 ├─ INI                  1            8            7            0            1
 ├─ Shell                1           20           20            0            0
 (Total)                             1342           27          936          379
===============================================================================
 Total                6898      2820183       975515      1702351       142317
===============================================================================
[mydev@fedora renode-master-with-submodules]$
```

- **Submodules**

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

```
[submodule "src/Infrastructure"]
        path = src/Infrastructure
        url = https://github.com/renode/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
[submodule "lib/bc-csharp"]
        path = lib/bc-csharp
        url = https://github.com/antmicro/bc-csharp.git
```

Hierarchy

```
[mydev@fedora renode-master-with-submodules]$ tree -L 2 .
.
├── ACKNOWLEDGEMENTS.rst
├── build.sh
├── CHANGELOG.rst
├── CONTRIBUTING.rst
├── lib
│   ├── AntShell
│   ├── bc-csharp
│   ├── BigGustave
│   ├── cctask
│   ├── CxxDemangler
│   ├── ELFSharp
│   ├── FdtSharp
│   ├── InpliTftpServer
│   ├── Migrant
│   ├── options-parser
│   ├── Packet.Net
│   ├── termsharp
│   └── xwt
├── LICENSE
├── platforms
│   ├── boards
│   └── cpus
├── README.rst
├── renode
├── Renode.sln
├── renode-test
├── scripts
│   ├── complex
│   ├── monitor.py
│   ├── multi-node
│   ├── pydev
│   └── single-node
├── src
│   ├── Infrastructure
│   ├── Plugins
│   └── Renode
├── tests
│   ├── metrics-analyzer
│   ├── network-server
│   ├── nunit_tests_provider.py
│   ├── peripherals
│   ├── platforms
│   ├── requirements.txt
│   ├── robot_output_formatter.py
│   ├── robot_output_formatter_verbose.py
│   ├── robot_tests_provider.py
│   ├── run_tests.py
│   ├── tests_engine.py
│   ├── tests.yaml
│   ├── tools
│   └── unit-tests
└── tools
    ├── building
    ├── common.sh
    ├── gdb_compare
    ├── metrics_analyzer
    ├── mono_version
    ├── packaging
    ├── sel4_extensions
    └── version
```

```
[mydev@fedora renode-master-with-submodules]$ tree -L 4 -d src/Infrastructure
src/Infrastructure
├── licenses
└── src
    ├── Emulator
    │   ├── Cores
    │   │   ├── Arm
    │   │   ├── Arm-M
    │   │   ├── Common
    │   │   ├── Debug
    │   │   ├── PowerPC
    │   │   ├── renode
    │   │   ├── RiscV
    │   │   ├── Sparc
    │   │   ├── tlib
    │   │   ├── X86
    │   │   └── Xtensa
    │   ├── Extensions
    │   │   ├── Analyzers
    │   │   ├── Backends
    │   │   ├── Config
    │   │   ├── Hooks
    │   │   ├── HostInterfaces
    │   │   ├── MonitorTests
    │   │   ├── TAPHelper
    │   │   ├── Tools
    │   │   ├── UserInterface
    │   │   └── Utilities
    │   ├── Main
    │   │   ├── Backends
    │   │   ├── Core
    │   │   ├── Debug
    │   │   ├── Exceptions
    │   │   ├── Foreign
    │   │   ├── Logging
    │   │   ├── Network
    │   │   ├── Peripherals
    │   │   ├── Plugins
    │   │   ├── Sound
    │   │   ├── Storage
    │   │   ├── Testing
    │   │   ├── Tests
    │   │   ├── Time
    │   │   ├── UserInterface
    │   │   └── Utilities
    │   └── Peripherals
    │       ├── Peripherals
    │       └── Test
    ├── Plugins
    │   ├── AdvancedLoggerViewerPlugin
    │   ├── SampleCommandPlugin
    │   └── TracePlugin
    │       └── Handlers
    └── UI
        ├── ConsoleBackendAnalyzers
        ├── VideoAnalyzer
        │   ├── Events
        │   └── Handlers
        └── XwtProvider
            └── Progress
```

```
[mydev@fedora renode-master-with-submodules]$ tree -L 4 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
│   │   └── src
│   │       ├── buses
│   │       ├── peripherals
│   │       ├── renode_bus.cpp
│   │       ├── renode_bus.h
│   │       ├── renode_cfu.cpp
│   │       ├── renode_cfu.h
│   │       ├── renode.h
│   │       └── verilator-integration-library.cmake
│   ├── VerilatorPlugin.cs
│   └── VerilatorPlugin.csproj
└── WiresharkPlugin
    ├── BLESniffer.cs
    ├── INetworkLogExtensions.cs
    ├── LinkLayer.cs
    ├── Wireshark.cs
    ├── WiresharkPlugin.cs
    ├── WiresharkPlugin.csproj
    └── WiresharkSender.cs
```

```
src/Infrastructure/src/Emulator/Cores/tlib
├── arch
│   ├── arm
│   ├── i386
│   ├── ppc
│   ├── riscv
│   ├── sparc
│   └── xtensa
│       ├── core-apollolake
│       ├── core-baytrail
│       ├── core-cannonlake
│       ├── core-dc233c
│       ├── core-de212
│       ├── core-de233_fpu
│       ├── core-dsp3400
│       ├── core-haswell
│       ├── core-icelake
│       ├── core-imx8
│       ├── core-imx8m
│       ├── core-sample_controller
│       ├── core-test_kc705_be
│       ├── core-test_mmuhifi_c3
│       └── core-tigerlake
├── fpu
├── include
└── tcg
    ├── arm
    └── i386
```

QEMU

- **Stats without submodules (last commit: a8cc5842b5cb863e46a2d009151c6ccbdecadaba)**

```
[mydev@fedora qemu-master-without-submodules]$ tokei
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 GNU Style Assembly    158        16923        13129         1305         2489
 Autoconf                3           13           13            0            0
 BASH                  231        29239        16528         7590         5121
 C                    3383      1741262      1343755       177875       219632
 C Header             2281       324155       199050        86821        38284
 C++                     4        24039        13712         6897         3430
 C++ Header              1           50           22           23            5
 CSS                     1          161           99           29           33
 Device Tree             4         1398         1196           79          123
 GLSL                    3           30           21            0            9
 Haxe                    5         8454         7723           11          720
 HEX                     1           18           18            0            0
 HTML                    1           14           10            1            3
 INI                     2           21           20            0            1
 JavaScript              1            9            7            1            1
 JSON                  253        27838        26656            0         1182
 LD Script               2           54           42            2           10
 Makefile              122         2134         1545          286          303
 Markdown                2          155            0          104           51
 Meson                 207        10850         9576          405          869
 Module-Definition      15         7523         6195            0         1328
 Objective-C             4         3353         2481          427          445
 Perl                    6         5600         4586          396          618
 Python                343        73456        53919         8502        11035
 ReStructuredText      214        40294        30099            0        10195
 Shell                  71        12338         9719         1646          973
 SVG                     2         1986         1982            2            2
 Plain Text             55        16142            0        13624         2518
 XML                    40         2626         1688          787          151
 YAML                    6          547          433           72           42
===============================================================================
 Total                7421      2350682      1744224       306885       299573
===============================================================================
[mydev@fedora qemu-master-without-submodules]$
```

Stats with submodules (last commit: d136ec13703f175fdf5eb94aa9b2c851570994a4)

```
[mydev@fedora qemu-master-with-submodules]$ tokei
```

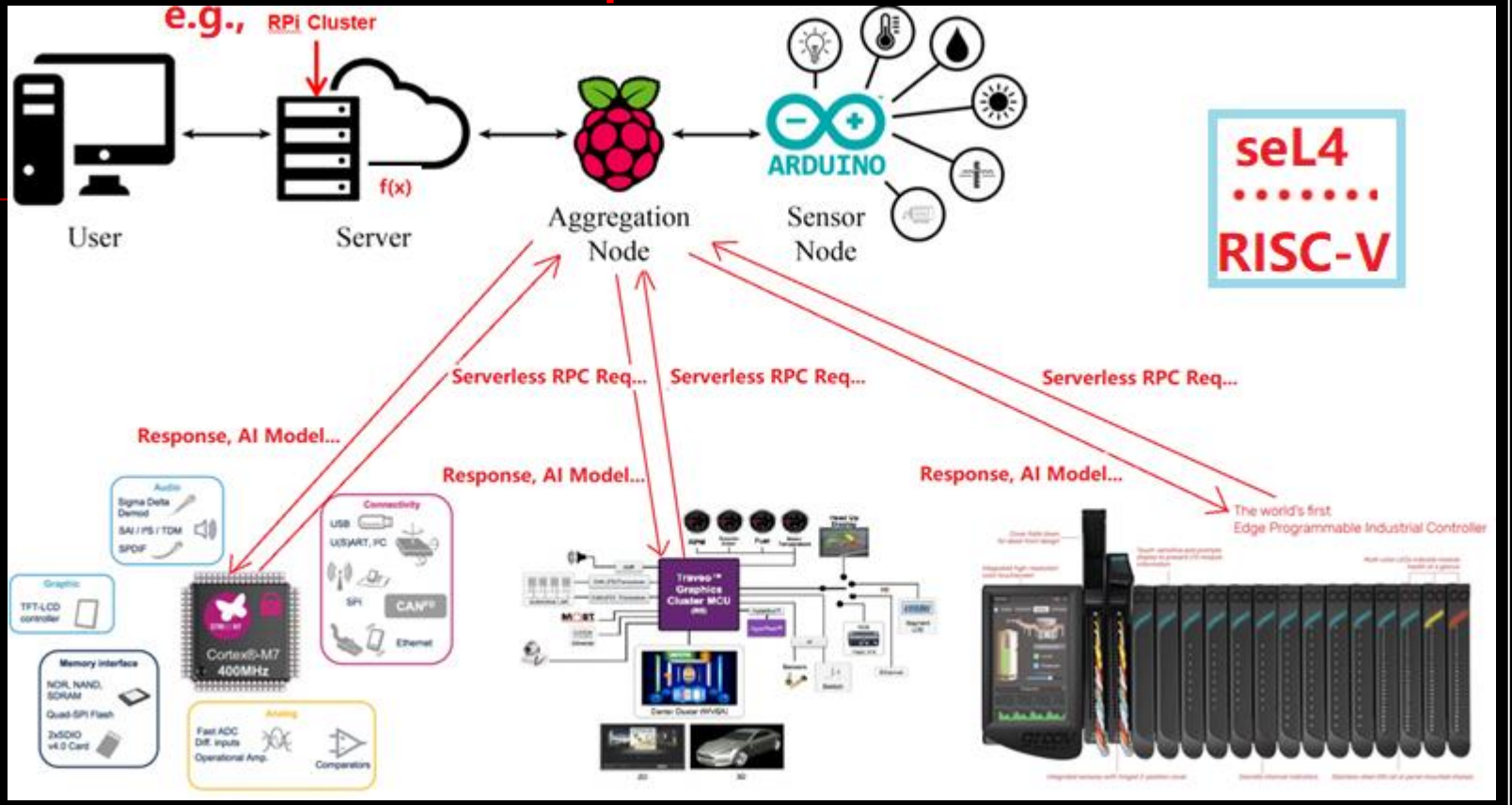
Language	Files	Lines	Code	Comments	Blanks
Alex	2	273	245	0	28
ASN.1	14	2507	1692	404	411
Assembly	101	10429	6144	2745	1540
GNU Style Assembly	843	176105	125548	28712	21845
Autoconf	375	26414	18688	4117	3609
Automake	21	1417	1070	65	282
BASH	328	33546	19821	8079	5646
Batch	96	18069	12796	2454	2819
C	19189	7846881	5544122	1363130	939629
C Header	13911	3002655	1800669	894427	307559
CMake	172	11535	7756	1909	1870
C#	47	9877	7698	1646	533
C Shell	1	154	147	4	3
Coq	1	32	28	0	4
C++	386	122776	87139	20042	15595
C++ Header	54	798	493	170	135
CSS	8	2591	2169	98	324
D	19	295	228	7	60
Device Tree	2797	573814	477465	32017	64332
Dockerfile	44	5682	5541	14	127
.NET Resource	1	31	31	0	0
Emacs Lisp	6	276	159	89	28
F#	268	39253	33588	142	5523
FORTRAN Legacy	2	19	13	0	6
FORTRAN Modern	47	678	461	15	202
GDB Script	3	356	207	93	56
GLSL	3	30	21	0	9
Go	53	32696	25871	3212	3613
Happy	9	4773	4176	0	597
Haxe	5	8454	7723	11	720
HEX	12	1580	1538	0	42
INI	24	1237	946	196	95
Java	119	19412	14549	3337	1526
JavaScript	20	9750	6806	2738	206
JSON	524	32951	31758	0	1193
LLVM	1	4	4	0	0
LD Script	139	11155	7358	2386	1411
Lua	2	156	124	2	30
Makefile	1651	59188	35429	14877	8882
Meson	1413	29494	23067	1930	4497
Module-Definition	34	8487	7048	69	1370
Objective-C	12	3407	2524	427	456
Objective-C++	4	41	29	0	12
Pan	1	49	33	3	13
Perl	370	281951	230948	23324	27679
PHP	26	4026	2664	700	662
PowerShell	2	128	88	18	22
Prolog	3	679	677	0	2
Protocol Buffers	6	74	61	0	13
Python	1799	529897	401315	55141	73441
R	2	1057	975	2	80
RPM Specfile	2	179	136	6	37
ReStructuredText	947	157678	117937	0	39741
Rust	31	212	166	5	41
Shell	370	55322	39548	8062	7712
SQL	1	90	85	0	5
SVG	16	4273	4242	15	16
Swift	11	56	45	0	11
SWIG	2	1376	1147	105	124
TCL	8	1645	1216	178	251
TeX	9	9605	7260	612	1733
Plain Text	871	569553	0	480755	88798
TOML	3	19	15	2	2
TypeScript	4	48	48	0	0
Vala	46	466	383	6	77
Vim script	4	358	307	33	18
Visual Studio Pro|	62	10587	10587	0	0
Visual Studio Sol|	8	853	847	0	6
XSL	8	811	576	96	139
Xcode Config	6	146	33	81	32
XML	96	5310	4082	901	327
YAML	213	15307	13513	951	843
HTML	132	20286	18395	49	1842
├─ CSS	4	20	20	0	0
├─ JavaScript	9	73	63	5	5
(Total)		20379	18478	54	1847
Markdown	303	53467	0	40006	13461
├─ Autoconf	1	28	28	0	0
├─ BASH	12	64	42	13	9
├─ C	9	165	133	22	10
├─ CMake	3	6	6	0	0
├─ C++	2	14	12	0	2
├─ INI	16	194	172	7	15
├─ JSON	4	144	144	0	0
├─ Meson	75	2205	1872	147	186
├─ Python	2	43	25	4	14
├─ Shell	2	7	4	2	1
├─ XML	1	17	13	4	0
├─ YAML	1	127	100	7	20
(Total)		56481	2551	40212	13718
Total	48123	13834786	9180218	3000615	1653953

```
[mydev@fedora qemu-master-with-submodules]$
```

VI. Wrap-up

- A New Golden Age for FOSS EDA!
- We do think  is one of the best candidate for next generation system programming, especially in the HW-SW co-designed Edge Computing Infrastructure which has more freedom to innovate.
-  has great potential, but still has a long way to go…
- You may look forward to our upcoming follow-ups "Revisiting D as a better system programming language" and "Rethinking D for HW-SW co-designed system" in this area.
- Continuously updating "Revisiting the eBPF-centric new approach for Hyper-Converged Infrastructure & Edge Computing" at K+ Summit 2021(Shanghai), and the third-round discussion on this topic will be divided into two series "ARM + Python + Rust + Lua + GraalVM + …" and "RISC-V + Python +  + Lua +  + …" according to different technology roadmap.

- **Overview of our solution plan**



Integrating eBPF, Python, D, Lua, Wasm, HPC, Ray based and more open-source projects for a novel lightweight infrastructure of Edge Computing & FOSS EDA.

Q & A

Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/
- https://en.wikipedia.org/wiki/GNU_Compiler_Collection
- https://en.wikipedia.org/wiki/Self-hosting_(compilers)
- https://en.wikipedia.org/wiki/Coroutine
- https://en.wikipedia.org/wiki/SystemVerilog
- https://en.wikipedia.org/wiki/Hardware_verification_language
- https://github.com/kennyalive/Language-Arena
- https://thume.ca/2019/07/14/a-tour-of-metaprogramming-models-for-generics/
- https://users.rust-lang.org/t/dlang-adds-a-borrowchecker-called-the-ob-system-for-ownership-borrowing/42872/11
- https://opensource.com/article/17/5/d-open-source-software-development
- https://scalameta.org/
- https://wiki.dlang.org/LDC_CUDA_and_SPIRV
- https://github.com/ben-marshall/awesome-open-hardware-verification
- https://github.com/hdl/bazel_rules_hdl
- https://blog.devgenius.io/d-c-evolution-programming-on-android-9bdf9aa1b67d

- https://wiki.radxa.com/Rock5/hardware/5b
- https://trends.google.com/trends/explore?date=all&q=D%20language,Rust%20language
- https://cacm.acm.org/magazines/2018/3/225475-a-programmable-programming-language/fulltext

---

- https://www.khronos.org/spir/
- …