GHDL and the economy of EDA FOSS

Tristan Gingold - FSiC 2019

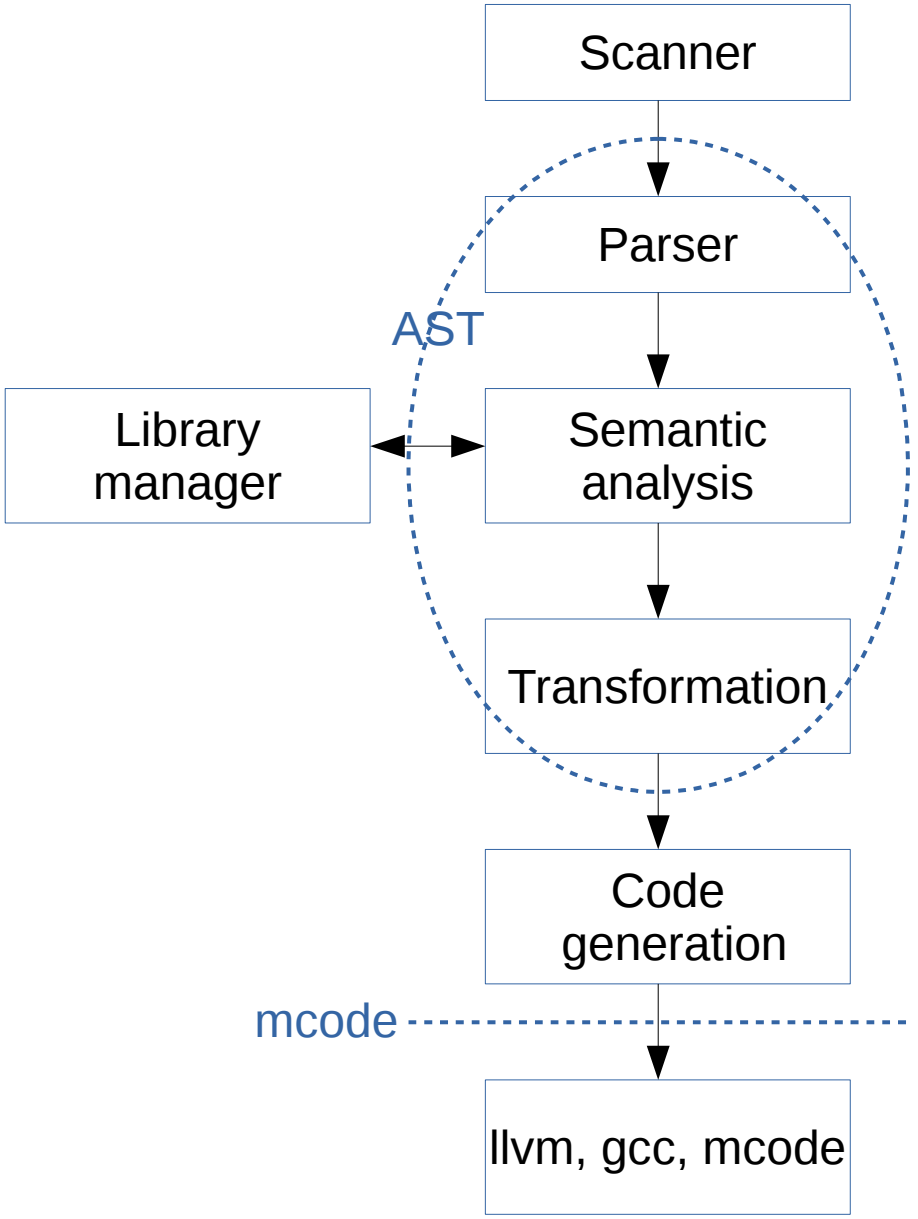
What is GHDL ?

- A Free (GPL v2+) VHDL simulator
- Supports IEEE 1076 (VHDL) 1987, 1993 and 2008 (partially)
- Compiled
- Command line tool
- Binaries for Linux (x86, x64), Windows, MacOS
- Generates VCD, GHW and FST waveforms
- VPI support (for external tools like cocotb)

What does it support ?

- Full support of VHDL-87 and 93
  - Your design should work
- Partial support of VHDL-2008
  - Good enough for VUnit, OSVVM and UVVM.
- Support inline PSL (assert and cover)
  - Creates an NFA

Internal view



- Generate tokens — scan.adb
- Generate the AST — parse.adb
- Create links in the AST
- Semantic checks — sem*.adb
- Rewrite as processes — canon.adb
- Generate a low-level AST — trans*.adb
- Generate object code

Elaboration

- Object code can be generated:
  - File per file (gcc or llvm backend)
    - No C generated, it is ghdl1
  - For the whole design (mcode – an internal JIT)
    - Very fast and lightweight, no opt
- Elaboration is dynamic
  - Done early during execution
  - Prevents many optimizations

Abstract Syntax Tree - AST

- Represent the sources
- Closely follow the VHDL Language Reference Manual
  - Simplify the support of the language
  - Possible as the LRM is well/correctly written
- Strongly typed
  - Each field of a node has a type

Abstract Syntax Tree - AST

- Nodes are represented by 32-bit numbers
  - Shorter than pointers on 64 bit platforms
- Nodes size is fixed (32 or 64 bytes)
  - Possible to implement parallel tables
  - To extend the nodes (eg: in translate)

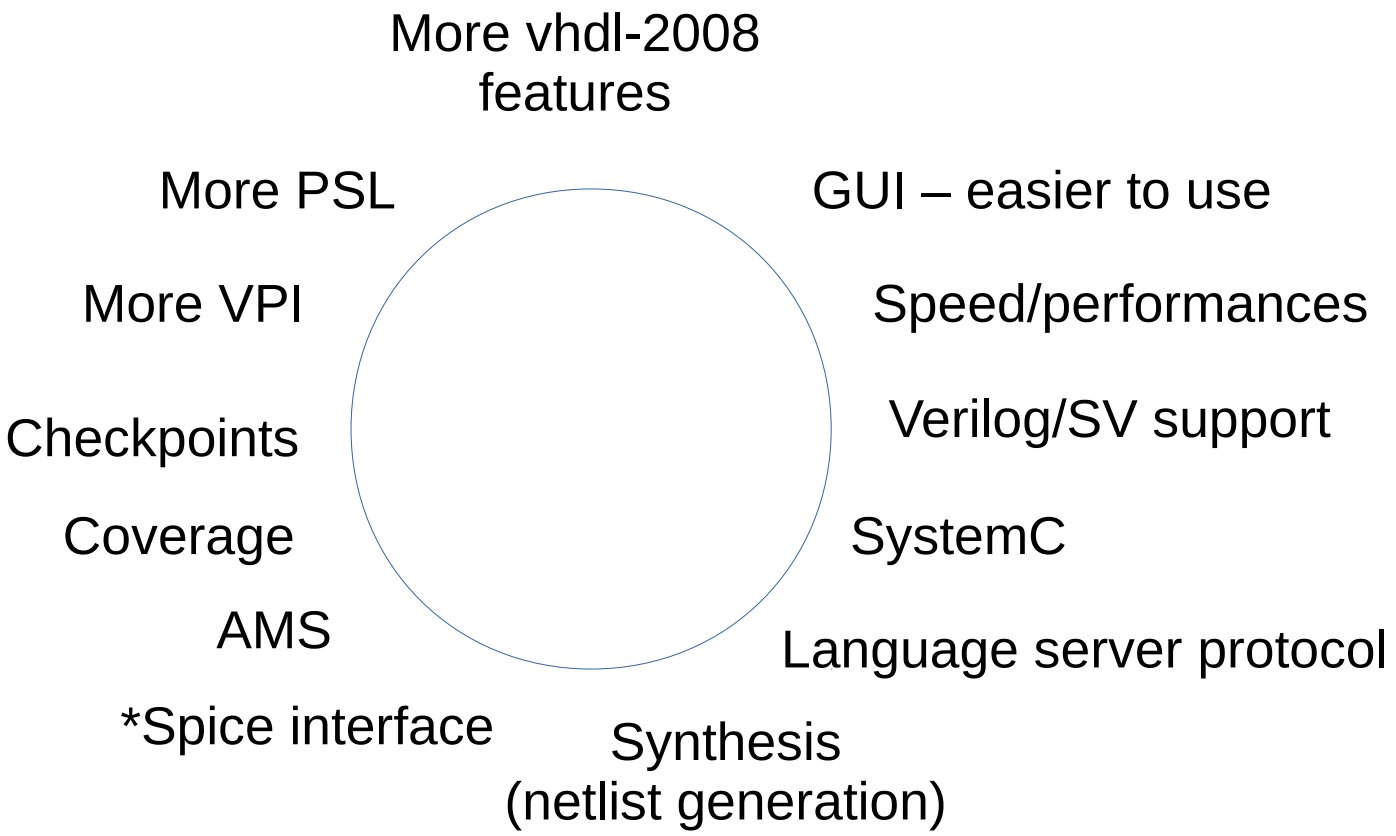
Abstract Syntax Tree - AST

- Meta-description
  - Type: node, list, number, flags, ...
  - Links: own, reference, maybe-own
  - Simplify package instantiation or dump.
- Described using ad-hoc comments

Why Ada ?

- GHDL is written in Ada
- A subset (imperative, sequential)
- Strong typing
- Modular language
- Stable
- VHDL is very similar to Ada
  - If you know one, you will be familiar with the other
- Makes contribution less easy ?

The future of GHDL



Collaboration: analog

- VHDL-AMS:
  - Parsing and analysing AMS extensions
  - DAE solver
- Spice/GnuCap
  - Linking simulation kernel together

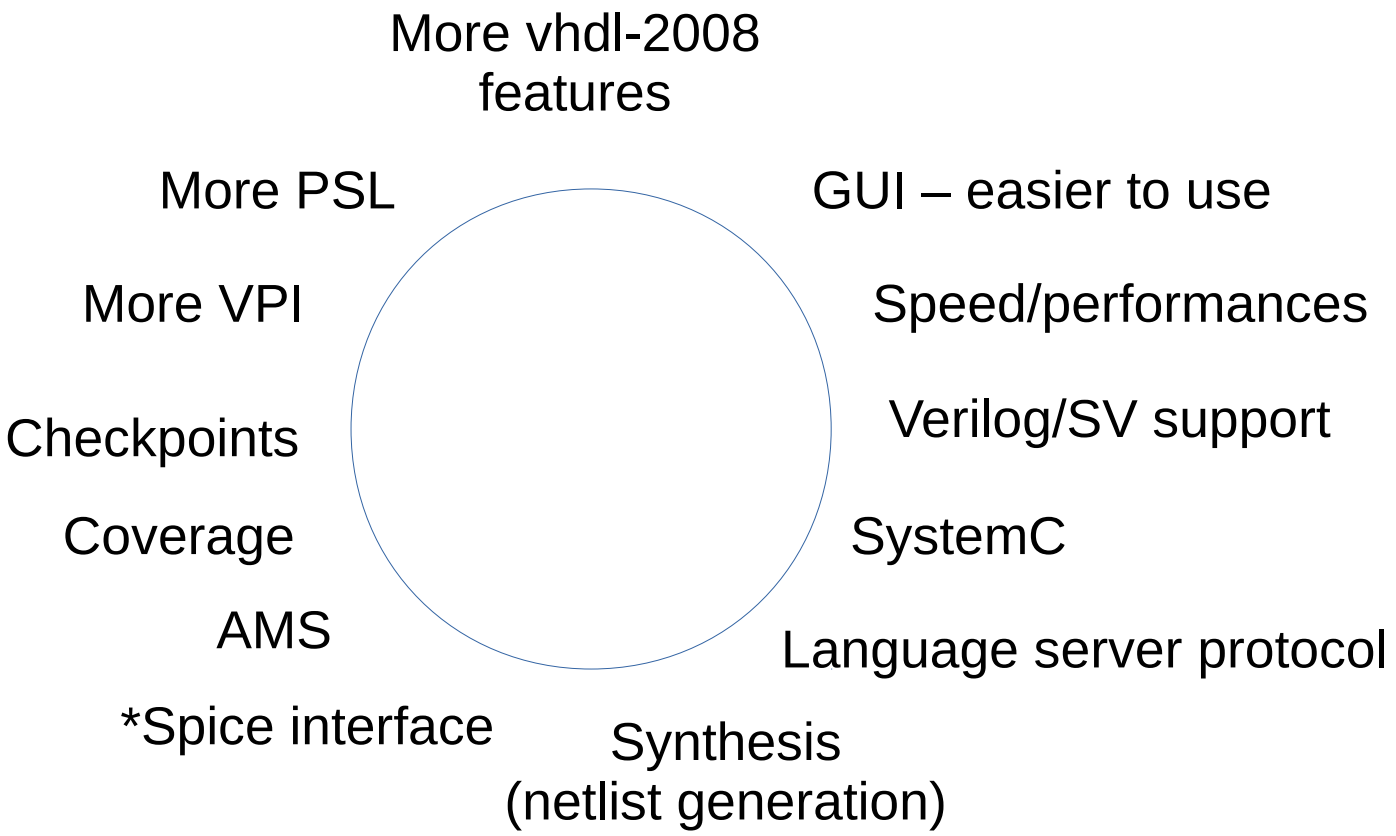
Collaboration: Synthesis

- Proof of concept (ghdlsynth)
  - Need to support arrays
  - Yosys module

Collaboration: SystemVerilog

- Mixed vhdl + verilog/systemverilog
  - Even support SystemC ?
- SystemVerilog
  - Start from scratch ?
  - Or start from an existing simulator ?

Priority ?


More vhdl-2008 features
GUI – easier to use
Speed/performances
Verilog/SV support
SystemC
Language server protocol
Synthesis (netlist generation)
*Spice interface
AMS
Coverage
Checkpoints
More VPI
More PSL

The economy of EDA FOSS

- My experience
- My feelings

Why writing FOSS EDA tool ?

- By philosophy
  - I want FOSS everywhere
- Academic research project
  - I need a tool for my research project or my PhD
- To understand a technology
  - I'd like to learn a language
- To improve my tooling
  - The tool is crappy
  - I need a specific feature

Audience / market

- Maybe 100x CS engineers than EE engineers
  - Do not expect a lot of users!
- Cost: you just need a computer to write an app, you need >40$ for a little EE project
- Culture: FOSS is SW first
  - Many HW designers are happy with closed source EDA

Contributions

- Skill: EE is not CS
  - HW people aren’t SW developers
  - SW people aren’t interested in HW design
- Tools are already available
  - Free HDL simulators, synthesis, P&R
  - Why working on a FOSS project ?
- Do not expect many contributions
  - YMMV ?

Complexity

- Lot’s of complex algorithms or data structure
  - BDD, SAT, NP-hard, simulated annealing…
- Makes EDA very interesting / challenging
- But not for beginners
- Need more time to tune an algorithm
- Reuse existing libraries
  - When possible
  - When existing

Closed ecosystem

- Most low-level information are not available
  - FPGA bitstream and characteristics
  - Foundry processes
  - IP cores are encrypted
  - File formats (schematic, waveforms…)
- Difficult to write a FOSS element

Maturity

- Prototype vs mature software
- Users need reliable and stable software
- Authors usually need a prototype
- Technology changes quickly
- Need to use industry file formats
- My experience:
  - every VHDL feature has been used
  - difficult to support only a subset