**<https://github.com/AAOleynikov/MVHDL> - репозиторий фронта**

[**https://aaoleynikov.github.io/MVHDL/**](https://aaoleynikov.github.io/MVHDL/)

[**https://todo.zhugeo.ru/projects/9/kanban**](https://todo.zhugeo.ru/projects/9/kanban)

[**https://github.com/AAOleynikov/MVHDL\_back**](https://github.com/AAOleynikov/MVHDL_back) **- репозиторий бэка**

**Стек технологий фронт: HTML+CSS+JS+VUE+NODE.JS+NPM+GITHUB+WEBPACK+WASM**

**Для JS фронт: Monaco IDE+antlr4+VCD.Drom+FancyTree+jquerry.ui**

**Нативное кросплатформенное приложение vs веб приложение** (WebAssembly позволяет раскрывать скорость нативных приложений + веб позволяет расширять кросплатформенность вплоть до умных часов :) + малый размер разрабатываемого функционала с точки зрения пользователя + примеры вроде VSCode на платформе Electron в формате веб-приложений и тд и тп…)

**Веб-приложение+сервер vs Статичное веб-приложение** (как таковая связь сервером может быть использована главным образом для трансляции, последующей компиляции и выполнения расчёта, но в ходе лаболаторных работ пользователь не пишет функционал, который не мог бы быть расчитан на его устройстве за приемлимое время + использование статичных веб-приложений позволяет воспользоваться Github pages, что в свою очередь *снимает любые затраты* на разворачивание и поддержку сервера) к сожалению этот вариант пока невозможен из-за сложности портирования на web GHDL/NVC, поэтому фронт на github pages, а запуск кода на сервере Георгия Жукова, с последующей возможностью на миграцию на сервер кафедры.

**Гайды по vhdl:**

<https://we.easyelectronics.ru/plis/vhdl-s-nulya.html>

Берчун Ю.В.Язык описания электронной аппаратуры VHDL

<https://github.com/matisiekpl/vhdl-runner> - VHDL Simulator, code VHDL in your browser! 🤖

<https://allaboutfpga.com/vhdl-testbench-tutorial/>

<https://technobyte.org/testbench-vhdl-types-examples-steps/>

**Доп. Инструментарий для web:**

<https://caniuse.com/wasm> - сборник функционала браузеров

**Примеры web-ide с хорошим UI, UX:**

**Основное UI/UX решение** основано на просмотре многих online IDE, среди многих было выявлено интересное -

<https://playcode.io/new> - js playground

<https://vscode.dev/> - web vscode

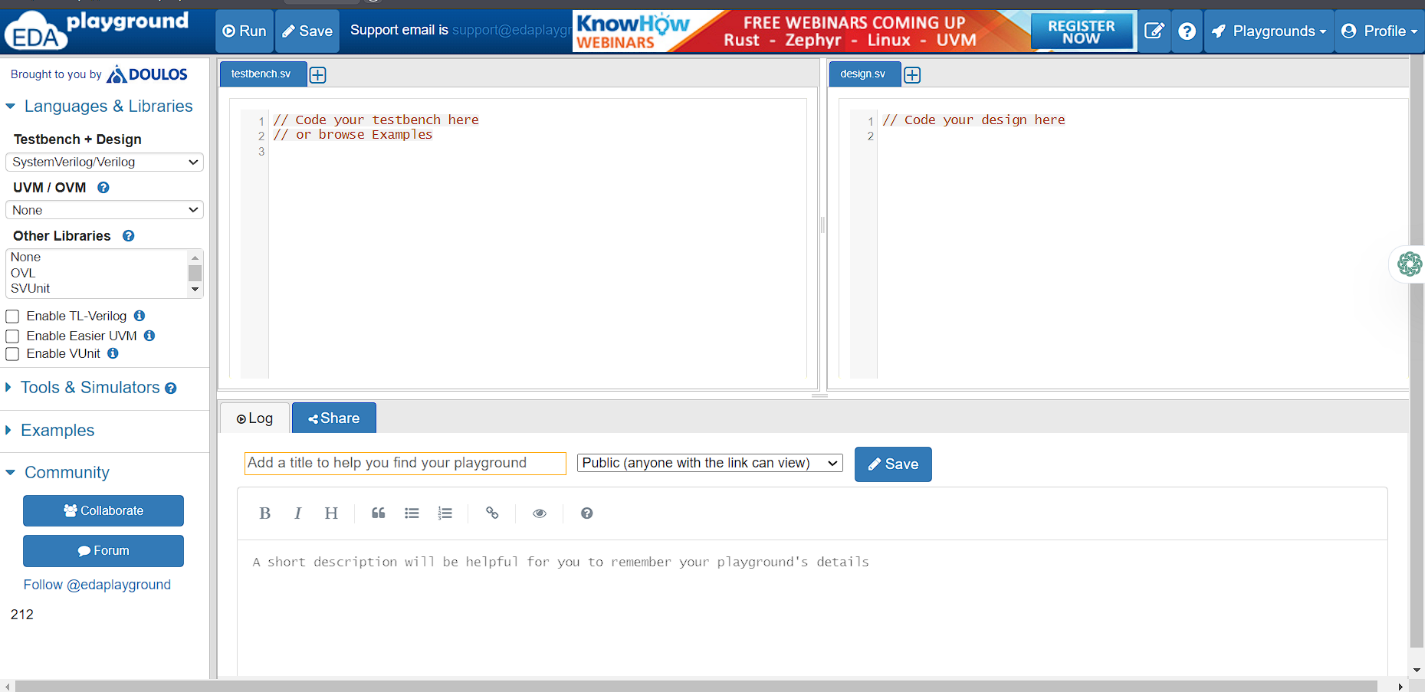
<https://www.edaplayground.com/> - поддерживает vhdl

<https://en.wikipedia.org/wiki/List_of_HDL_simulators>

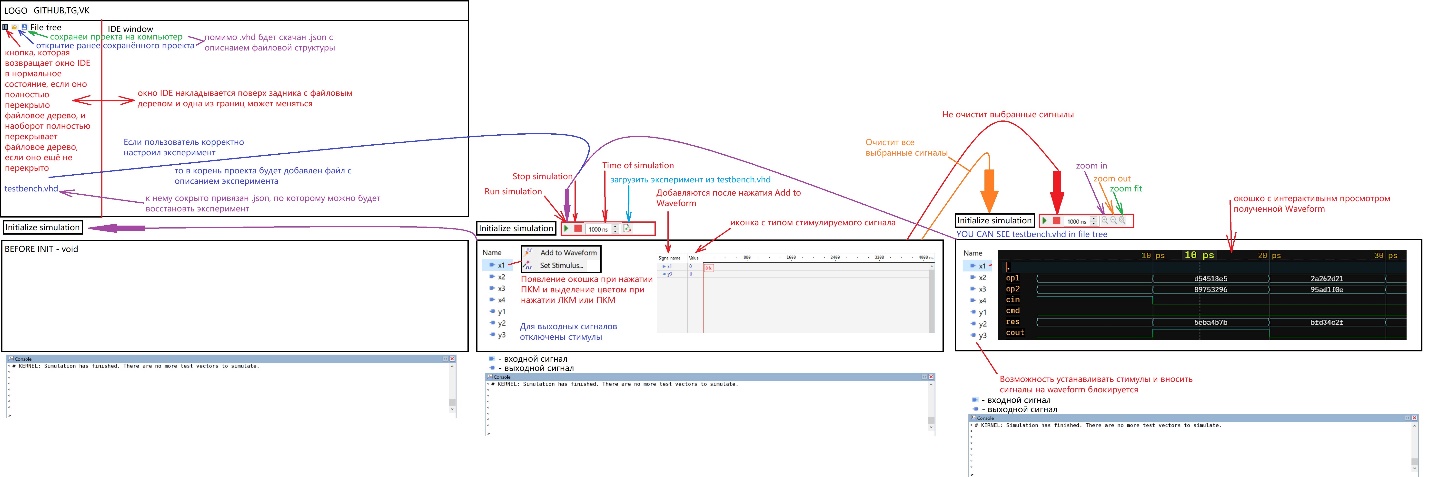
Не web-IDE рассмотренные – ACTIVE-HDL

На основании требований пользователя был разработан дизайн, поддерживающий необходимый функционал

Принципиальное отличие от ближайшего конкурента – EDA-playground наличие TESTBENCH генератора



Старая версия дизайна -



Новая – на доработке ЖУГЕО

**Материалы по MONACO-IDE:**

**Рассмотренные варианты:** Monaco IDE; Ace; CodeMirror

**Причины выбора:** (Открытый исходный код, MIT лицензия, активное сообщество, огромный встроенный функционал, лёгкий вес (10КБ??), встроенный функционал для разработки подсветки синтаксиса)

**Подсветка синтаксиса: TextMate** (тяжеловесный, требуется ломать монако, чтобы его подключить, требует Oniguruma) **vs Monarch** (встроен в монако, легковесный, но с низкой глубиной анализа (для VHDL полностью хватает т.к. язык по сути требует только подсветку ключевых слов))

<https://www.youtube.com/watch?v=ZJyp_tGsZ7I&t=1s> – яндекс про монако для внутренних интерфейсов

<https://habr.com/ru/companies/Voximplant/articles/445390/> - “IDE нормального человека или почему мы выбрали Monaco”

https://dev.to/coderpad/developer-diaries-how-we-built-a-better-browser-based-ide-with-monaco-5cfa - “Developer Diaries: How We Built a Better Browser-Based IDE with Monaco”

**Файловое дерево для навигации**

**FancyTree, jsTree, jqTree, Bootstrap Treeview, jQuery file tree, wunderbaum (beta), vue-tree-list**

+ FancyTree за скорость и функциональность + наибольшую распространённость

- jquerry ui

**Материалы по VUE:**

**Фреймворк** <https://habr.com/ru/companies/yandex_praktikum/articles/533702/>

<https://habr.com/ru/articles/329452/>

**WASM**

<https://github.com/jprendes/emception> - онлайн версия enscr

[https://habr.com/ru/articles/454868/](https://habr.com/ru/articles/454868/-) - Компиляция C в WebAssembly без Emscripten

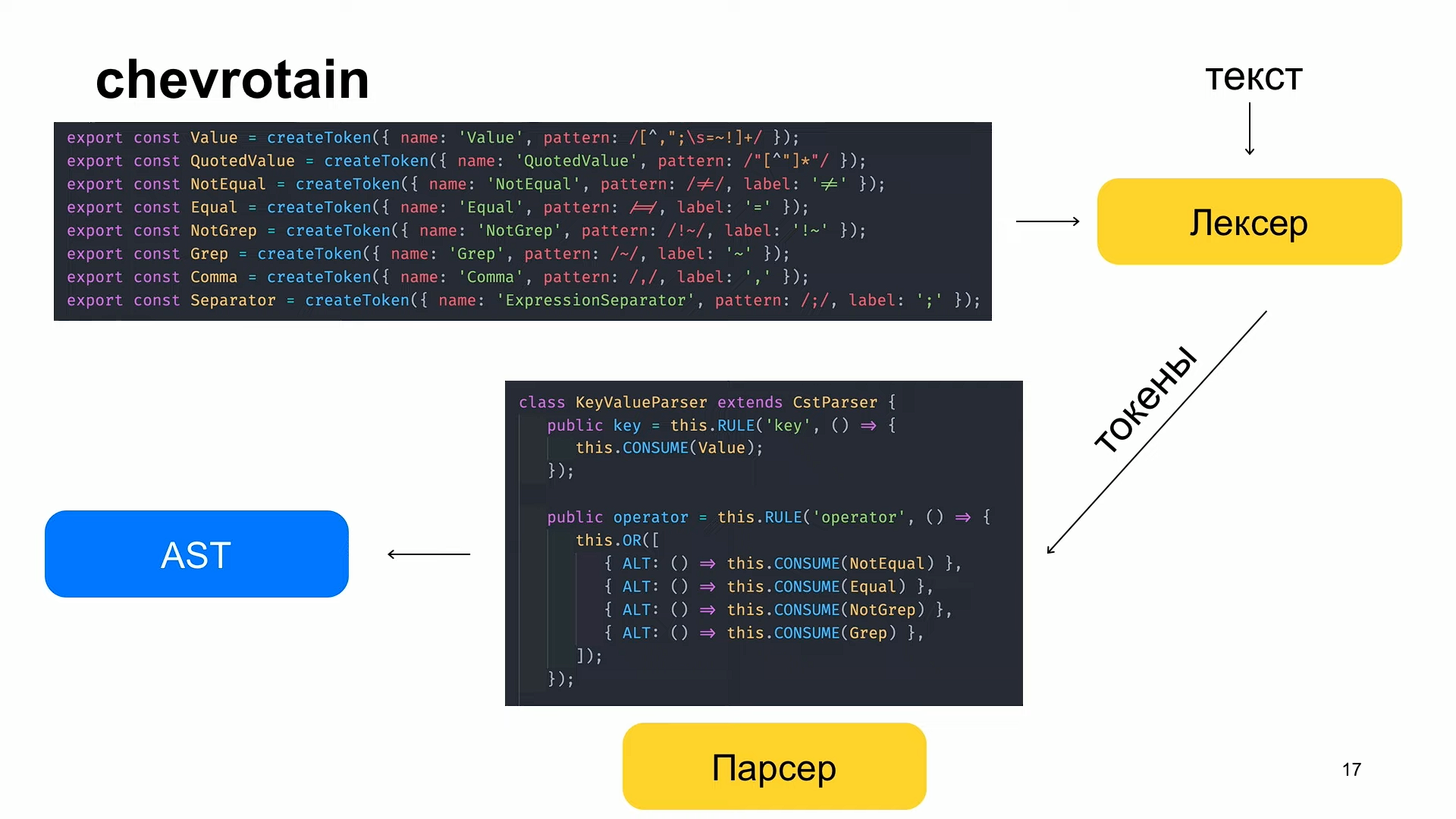
**Парсер сигналов из top-lvl VHDL .vhd файла**

**Рассмотренные варианты:** ANTLR (LL), chevroltain, LEX+YACC (LALR)

<https://chevrotain.io/playground/>

<https://github.com/antlr/antlr4>

VHDL – контекстно свободный **и возможен для разбора с помощью LL, LR, LALR????**



Яндекс использует chevroltain, но мы ANTLR за кроссплатформенность (протестировать .g4->C->WASM)

<https://github.com/popas90/vhdl-parser/tree/master> - popas90 VHDL parser

Написан кривой парсер на LEX, доработать и протестировать (проблемы: динамическая память, в файле yy.lex.c yymalloc, yycalloc, yyrealloc, yyfree должна быть вручну перегружена под WASM)

**Почему LEX вызывает только yymalloc, но не вызывает yyfree??**

<https://github.com/Nic30/VHDL_and_SV_formal_grammar_to_ANTLR4> -

-> можно использовать yacc + lex код которых сначала транслируется в код на Си, а потом в исполняемый файл на WEB-Assembly -> проблема совметсимости с устеравшими браузерами (IE даже новый не поддерживает)

\*\*\*почитать на странице GHDL в гитхабе как у них реализован лексер для

<https://tomassetti.me/antlr-mega-tutorial/#chapter35> - The ANTLR Mega Tutorial

<https://github.com/WGrape/lexer?tab=readme-ov-file> - A lexical analyzer based on DFA that is built using JS and supports multi-language extensions

<https://edautils.com/DownloadLinks.html>

***$ antlr4 -Dlanguage=JavaScript vhdlLexer.g4 vhdlParser.g4***

**Testbench generator**

<https://github.com/connorcl/testbench-gen> - A VHDL test bench generator for combinational and sequential logic, written in Python

<https://vhdl.lapinoo.net/testbench/> - Online VHDL Testbench Template Generator

<https://www.doulos.com/knowhow/perl/vhdl-testbench-creation-using-perl/> - VHDL Testbench Creation Using Perl

<https://www.itdev.co.uk/content/vhdl-testbench-generator-example>

<https://www.edautils.com/VHDLTBGen.html>

**Отображение сгенерированной Waveform**

https://en.wikipedia.org/wiki/Value\_change\_dump - VCD

<https://vc.drom.io/?github=AdoobII/idea_21s/main/vhdl/idea.vcd>

<https://github.com/wavedrom/wavedrom>

<https://gtkwave.sourceforge.net/> - десктоп приложение для просмотра waveforms

**Портирование GHDL на WEB**

<https://github.com/ghdl/ghdl> - This directory contains the sources of GHDL, the open-source analyzer, compiler, simulator and (experimental) synthesizer for VHDL, a Hardware Description Language (HDL).

Full support for the 1987, 1993, 2002 versions of the IEEE 1076 VHDL standard, and partial for the 2008 and 2019 revisions.

<https://github.com/eiselekd/vhdl2js>

<http://forum.pascal.net.ru/index.php?showtopic=32108> – форум про трансляторы с ADA на C/C++

<https://www.youtube.com/watch?v=nGM4FlSi7fM&t=482s> - про транслятор adaMagic (по итогам ресёрча стандарнтый GNAT в разы лучше)

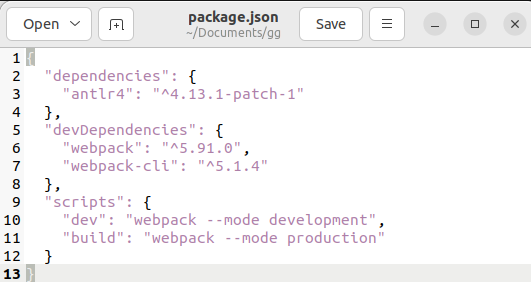
<https://blog.adacore.com/use-of-gnat-llvm-to-translate-ada-applications-to-webassembly> - Using GNAT-LLVM to target Ada to WebAssembly

<https://github.com/ghdl/docker>

**Портирование NVC на WEB**

<https://github.com/nickg/nvc/tree/master> - Меньший функционал, чем у GHDL, зато на C написан

**NPM**



***$ npm run build***

***$ cd dist; node main.js***

**НИРС по теме** “Разработка веб-интегрированной среды разработки (IDE) для логического синтеза, верификации и моделирования программ, основанной на языке описания аппаратуры (HDL) VHDL”.

**Содержание**

**Введение**

Развитие вычислительной техники привело к разработке языков описания цифровой аппаратуры. На сегодняшний день одними из основных языков описания цифровой аппаратуры являются Verilog (Verilog-HDL) и VHDL (***V****HSIC (Very high speed integrated circuits)* ***H****ardware* ***D****escription* ***L****anguage*) — язык описания аппаратуры интегральных схем.)

Современные среды разработки цифровой аппаратуры предоставляют широкий функционал моделирования и верификации проектов, где ввод устройств возможен не только с помощью языков описания аппаратуры, но и с помощью структурных схем. Наиболее качественные представители этих сред являются проприетарными, что усложняет процесс обучения студентов в условиях санкционной политики.

**Цель и задачи**

Цель данной работы:

изучить возможности применения ИНС в прикладных областях. Задачи данной работы: - анализ литературы о реализации и практическом применении ИНС; - обобщение полученной информации о предметной области, оформленное как теоретическая часть данной работы, и включающее в себя: - принцип работы; - классификацию; - процесс обучения; - практические примеры применения; - достоинства и недостатки; - практическая реализация трёхслойной ИНС, включая её обучение; - экспериментальное применение ИНС для решения задачи прогнозирования; - составление вывода на основе выполнения работы.

**Теоретическая часть**

**https://cyberleninka.ru/article/n/vhdl-i-verilog-hdl-yazyki-opisaniya-tsifrovoy-apparatury/viewer**

**Практическая часть**

**Общее описание работы**

В рамках описанной работы было реализовано web-IDE c бэком-компилятором

**Программное и техническое обеспечение**

**Серверная часть**

**\*описание сервера ЖУГЕО\***

**Использование библиотек, фреймвокров и технологий**