УДК 004.04

ПРОГРАММНАЯ ПОДСИСТЕМА ТЕСТИРОВАНИЯ ЗНАНИЙ ЯЗЫКОВ ОПИСАНИЯ АППАРАТУРЫ

С.В. Астахов fzastahov@gmail.com
Н.В. Лапшин nikita.lapshin2000@gmail.com
Т.А. Ким kimta@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Статья посвящена разработке программной подсистемы тестирования знаний языков описания аппаратуры, которая предоставляет возможности по управлению учебными материалами и автоматической проверке заданий, в том числе, заданий на описание аппаратных устройств на языке Verilog. Проведен анализ существующих систем тестирования знаний, в ходе анализа сформулированы функциональные требования и составлена диаграмма вариантов использования программной подсистемы тестирования знаний языков описания аппаратуры. Спроектирована архитектура и компоненты подсистемы. Проведено функциональное и нагрузочное тестирование разработанной подсистемы.

Ключевые слова

Тестирование знаний, язык описания аппаратуры, HDL, Verilog, система дистанционного обучения, образовательный портал.

Введение. В настоящее время существует огромное количество образовательных ресурсов, посвященных тематике информационных технологий. Несмотря на это, на данный момент в открытом доступе наблюдается дефицит ресурсов, посвященных изучению языков описания аппаратуры. Среди существующих образовательных платформ есть лишь несколько таких, на которых возможно настроить автоматическую проверку заданий на написание исходного кода на языке Verilog (или каком-либо другом языке описания аппаратуры) непосредственно в рамках веб-

приложения. При этом процесс настройки весьма сложен, поэтому авторы курсов редко используют описанную возможность.

В настоящей статье рассмотрен процесс разработки программной подсистемы тестирования знаний языков описания аппаратуры, которая предоставляет возможности по управлению учебными материалами и автоматической проверке заданий (в том числе, заданий на описание аппаратных устройств на языке Verilog).

Анализ существующих систем тестирования знаний. В ходе анализа существующих систем тестирования знаний, таких как Huawei University, Coursera, Stepik и Moodle авторами была предложена классификация методов тестирования знаний (таблица 1), составленная на основе классификации методов тестирования знаний, применяемых в системе Moodle [1].

Таблица 1

Классификация методов тестирования знаний

№	Тип	Подтип
1	Тестирование с ответом в закрытой форме	1.1 Выбор одного ответа 1.2 Выбор множественных ответов 1.3 Сопоставление
2	Тестирование с коротким ответом	2.1 С автоматизированной проверкой 2.2 С проверкой преподавателем 2.3 С перекрестной проверкой
3	Тестирование с ответом в форме эссе	3.1 С проверкой преподавателем 3.2 С перекрестной проверкой
4	Тестирование на написание исходного кода	4.1 С проверкой по референсным значениям 4.2 Автоматизированное тестирование на проверяющей стороне 4.3 Другие

В последующем, из рассмотренных методов тестирования знаний знаний были выделены наиболее подходящие для использования в подсистеме тестирования знаний языков описания аппаратуры методы. Кроме того, были предложены типы обратной связи, предоставляемой обучающемуся, в случае допущения им ошибки при решении задания (таблица 2).

Таблица 2

Методы тестирования знаний в разработанной подсистеме

№	Тип	Подтип	Вид обратной связи
1	Тестирование с ответом в закрытой форме	Выбор одного ответа	Текстовое пояснение ошибки
		Выбор нескольких ответов	Информации о наличии ложноположительных (ложноотрицательных) ответов
2	Задание на написание исходного кода	Автоматизированное тестирование на проверяющей стороне	Информация о несоответствующих сигналах

Кроме того, в результате проведенного анализа была составлена диаграмма вариантов использования подсистемы тестирования знаний языков описания аппаратуры, представленная на рисунке 1 [2].

Рис. 1. Диаграмма вариантов использования подсистемы

Проектирование архитектуры подсистемы. После анализа аналогов необходимым этапом является разработка архитектуры. Разработанную подсистему предполагается использовать как информационной системы образовательного портала, что отражено в архитектуре информационной системы, показанной на контекст-диаграмме (нотация C4) на рисунке 2 [3].

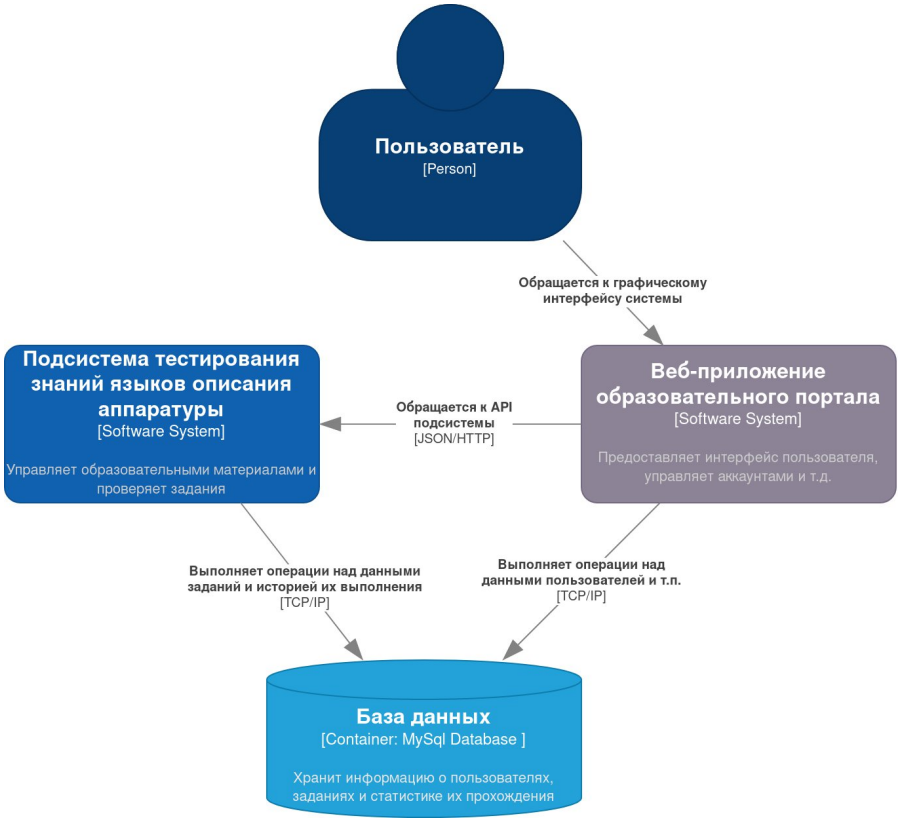

Рис. 2. Контекст-диаграмме информационной системы

Так как разработанная подсистема является весьма сложной, выполняемые в ней операции разнородны, могут потребовать использования различных языков и библиотек, а также некоторые из них могут занимать значительное время, при разработке было решено использовать микросервисную архитектуру [4].

На основе функциональных требований и представленной ранее диаграммы вариантов использования была разработана структура компонентов подсистемы, включающая следующие микросервисы:

- микросервис взаимодействия с БД — реализует CRUD-операции над данными в БД;
- микросервис анализа решений (анализатор) — выполняет проверку и анализ пользовательских решений;
- микросервис синтеза устройств (синтезатор) — выполняет синтез устройств из Verilog-кода и симулирует их работу;

– микросервис разбора временных диаграмм, микросервис генерации временных диаграмм wavedrom — преобразуют временные диаграммы в удобные для хранения и обработки форматы;
– микросервис анализа статистики;
– основной микросервис — реализует верхнеуровневую логику подсистемы, связывает остальные микросервисы.

Детализированная архитектура разработанной подсистемы показана на контейнер-диаграмме (нотация C4) на рисунке 2.

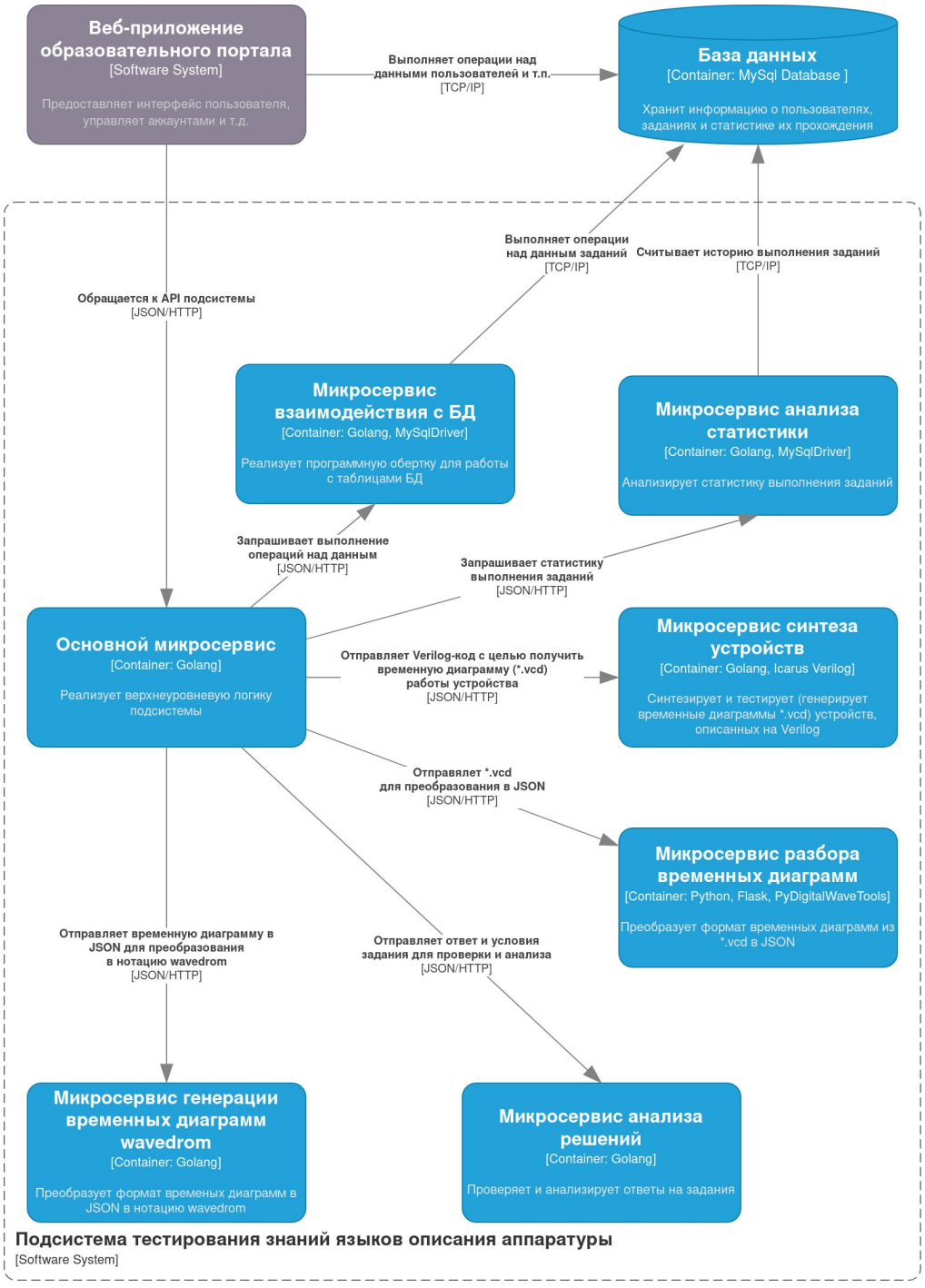

Рис. 3. Контейнер-диаграмма разработанной подсистемы

Проектирование базы данных. После проектирование архитектуры подсистемы была спроектирована структура ее базы данных. В результате анализа предметной области удалось выделить описанные ниже сущности [5].

Сущность "Задание" — содержит информацию о порядковом номере задания, его условиях, правильном ответе, цене в баллах и т.п.;

Сущность "Пользователь" — позволяет идентифицировать пользователя по ID, узнать, обладает ли пользователь правами администратора и узнать его псевдоним (т.н. "никнейм"). Кроме того, эта сущность может нести в себе дополнительную информацию, необходимую веб-приложению образовательного портала.

Сущность "Попытка решения" — содержит информацию, об успешности и времени каждой попытки решения задания каким-либо пользователем.

Полученная даталогическая схема базы данных в нотации Мартина изображена на рисунке 4.

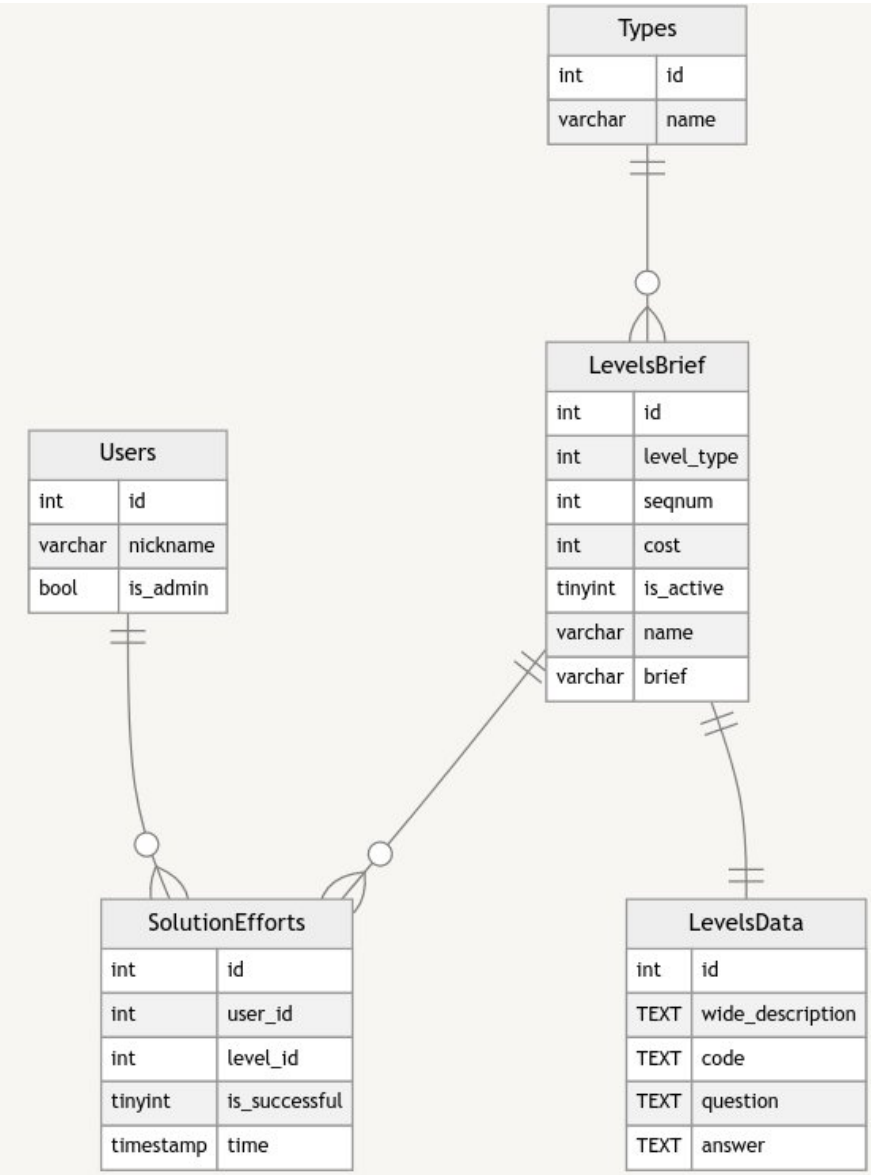

Рис. 4. Даталогическая схема базы данных

Разработка микросервиса синтеза устройств. Так как объем статьи не позволяет подробно рассмотреть процесс проектирования всех микросервисов, входящих в подсистему, ниже будет описан только микросервис синтеза устройств и микросервисы для работы с временными диаграммами.

Микросервис синтеза устройств по своей сути является оберткой вокруг программы Icarus Verilog, позволяющей моделировать работу цифровых устройств, описанных на языке Verilog. Программная обертка позволяет передавать в Icarus Verilog по сети исходный код, отправленный обучающимся в качестве ответа на задание в образовательном портале. Достоинствами Icarus Verilog являются [6]:

- малый размер исполняемого файла;
- наличие консольного режима работы (удобно вызывать из программного кода через библиотеки для работы с операционной системой);
- распространение по свободной лицензии (GNU GPL).

Обработка ответа обучающегося осуществляется в несколько этапов, за каждый из которых отвечает свой программный компонент:

- получение HTTP-запроса на симуляцию устройства (класс Router);
- сохранение полученных исходных кодов устрйоства и теста в файловой системе (OsLib, наличие user_id и level_id позволяет значительно снизить риск коллизии файлов);
- получение временной диаграммы работы устройства (в формате *.vcd) с помощью IcarusVerilog;
- удаление временных файлов;
- отправка HTTP-ответа, содержащего код временной диаграммы.

Диаграмма компоновки микросервиса показана на рисунке 5.

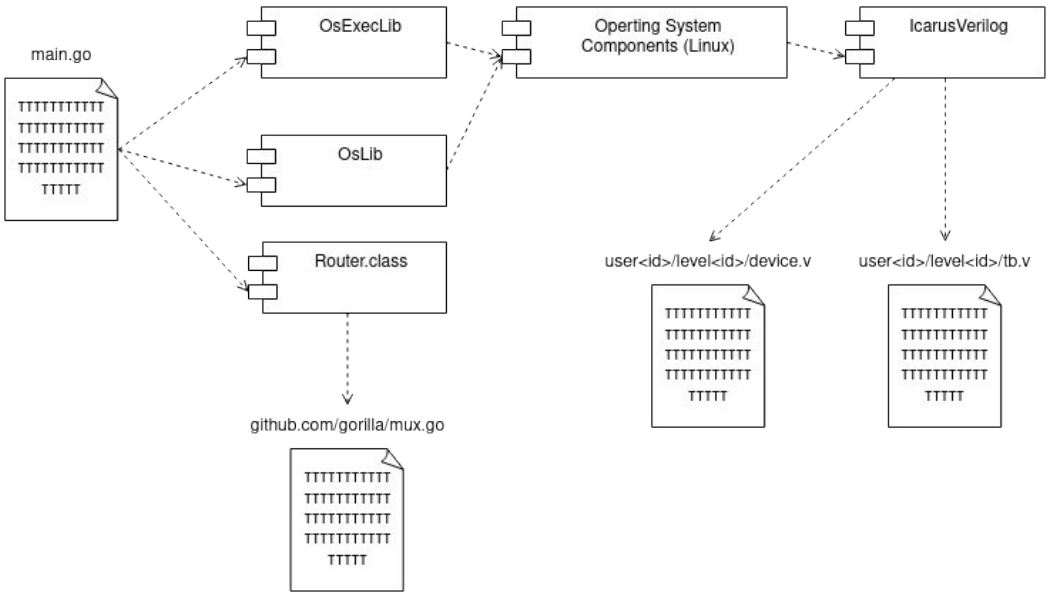

Рис. 5. Диаграмма компоновки микросервиса синтеза устройств

Микросервисы для работы с временными диаграммами. Для преобразования временных диаграмм в подсистеме предусмотрено два микросервиса: микросервис разбора временных диаграмм и микросервис генерации диаграмм wavedrom.

Изначально микросервис синтеза устройств в ходе тестирования работы устройства формирует временную диаграмму в формате *.vcd (Приложение Е). Данный формат крайне неудобен, как для анализа в сравнении с эталонной временной диаграммой, так и для генерации графического представления временной диаграммы в рамках веб-приложения.

Для преобразования временных диаграмм к более удобному для дальнейшей обработки формату был реализован микросервис разбора временных диаграмм.

Его исходный код написан на Python с применением библиотеки PyDigitalWaveTools [7]. Данная библиотека преобразует временную диаграмму в формате *.vcd в формат PyDigitalWaveTools согласно алгоритму, заложенному разработчиками библиотеки. Диаграмма Джексона, описывающая этот формат представлена на рисунке 6.

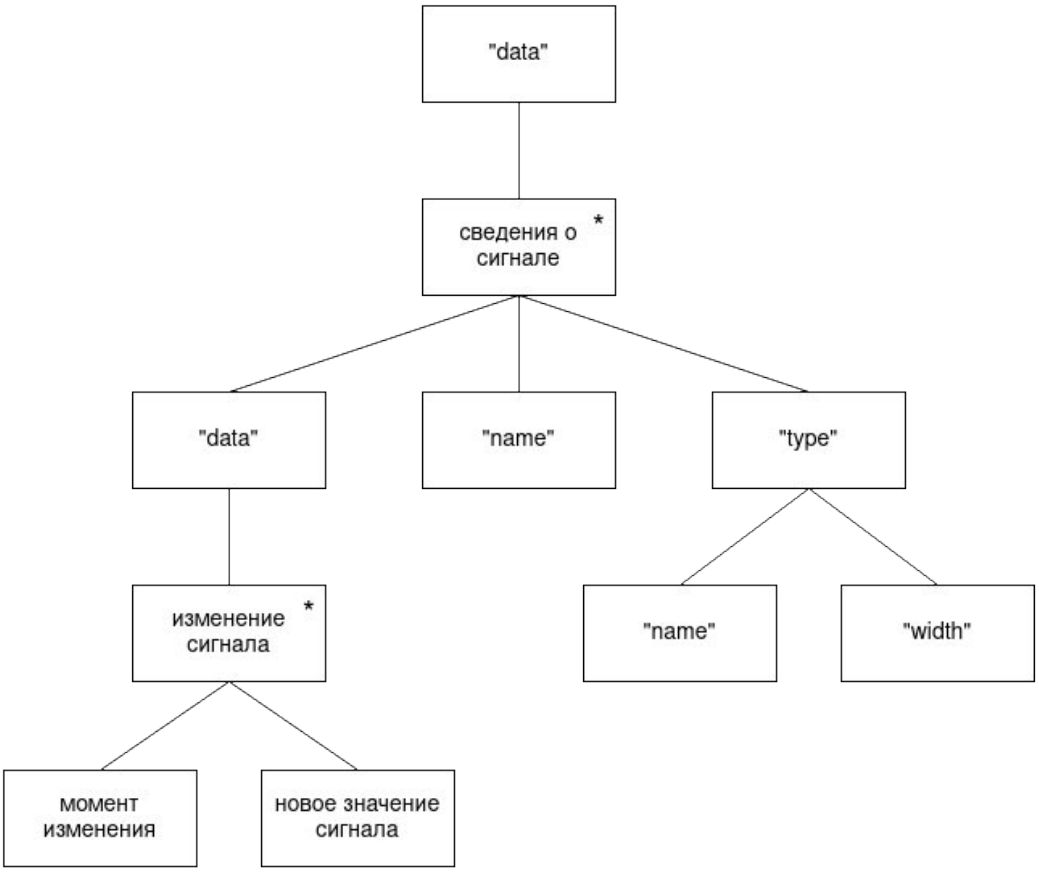

Рис. 6. Формат временных диаграмм в PyDigitalWaveTools

Формат PyDigitalWaveTools намного более удобен для сравнения с эталонной временной диаграммой (в том же формате) и анализа несоответствий, однако алгоритм визуализации для этого формата пришлось бы реализовать самостоятельно.

Вместо этого было решено реализовать микросервис генерации временных диаграмм wavedrom, который преобразовал бы временные диаграммы из формата PyDigitalWaveTools в формат движка Wavedrom [8]. Данный движок позволяет визуализировать временные диаграммы посредством http-запроса, содержащего описание сигнала, к специальному интернет-сервису.

Описание формата для движка Wavedrom в нотации Джексона приведено на рисунке 7.

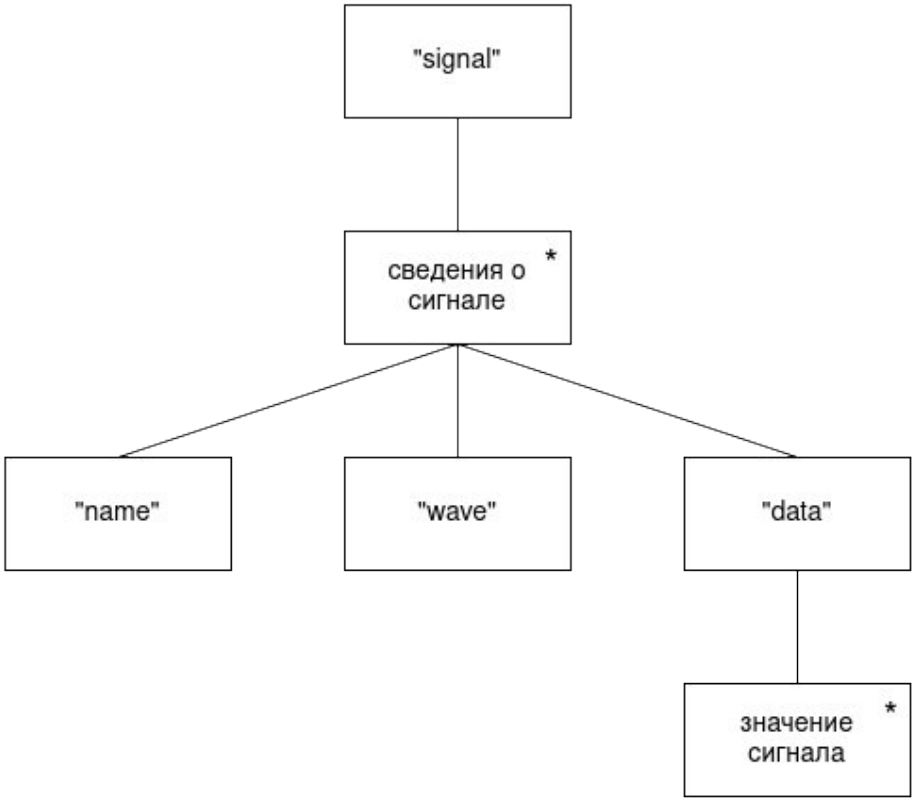

Рис. 7. Формат временных диаграмм для движка Wavedrom

Поля структуры имеют значение, описанное ниже:

- signal — массив всех сигналов временной диаграммы;
- name — имя сигнала;
- wave — форма сигнала (для каждого такта может иметь значения: "0", "1", "x", "z", "." — сохранить предыдущее, "|" — разрыв, "=" — обратиться к очередному элементу "data");
- data — массив, содержащий строковые значения сигнала (можно, например, отобразить большое число для многоразрядной шины).

Суть алгоритма преобразования из формата PyDigitalWaveTools в формат движка Wavedrom состоит в том, чтобы найти наибольший общий делитель для моментов изменения сигналов ($t_{НОД}$) и затем провести "дискретизацию" сигналов по времени, просматривая, как изменялся каждый сигнал в моменты времени кратные $t_{НОД}$.

Тестирование. После проектирования и реализации подсистемы тестирования знаний языков описания аппаратуры было проведено ее функциональное и нагрузочное тестирование.

Функциональное тестирование проводилось по принципу "черного ящика", для автоматизации тестирования применялась библиотека Pytest [9]. Для создания информативных отчетов о прохождении тестов была использована библиотека Allure [10]. Все 46 функциональных тестов прошли успешно, отчет о прохождении тестов представлен на рисунке 8.

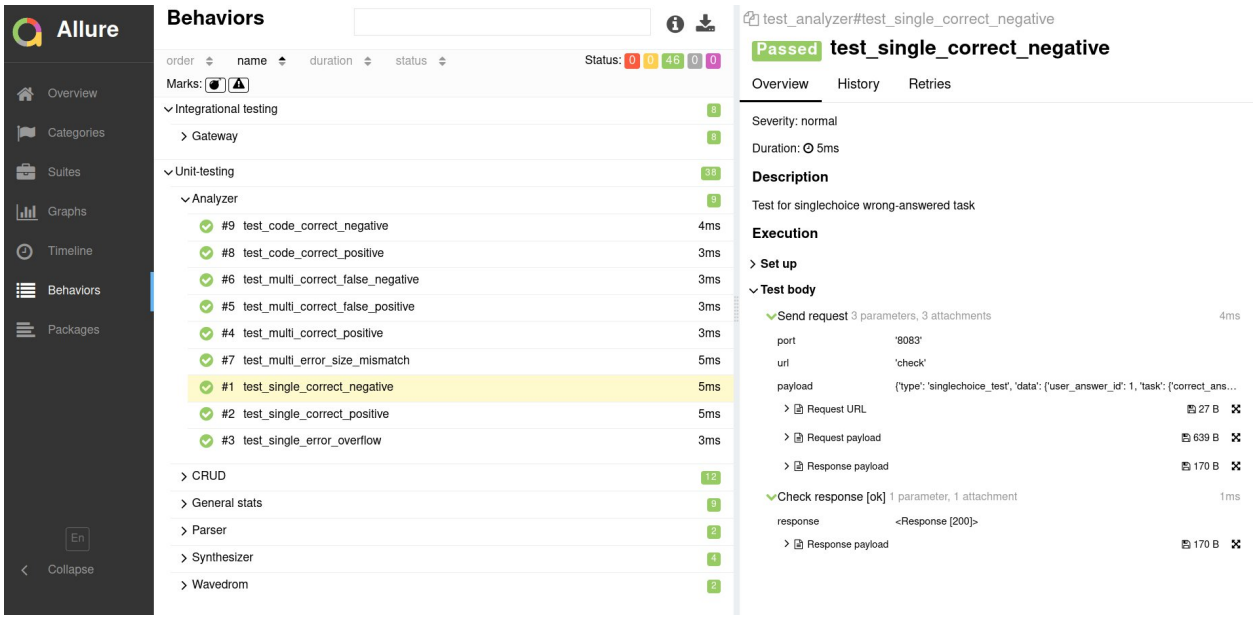

Рис. 8. Отчет о результатах функционального тестирования

Кроме того, было проведено нагрузочное тестирование основного микросервиса, микросервиса синтеза устройств и микросервиса разбора временных диаграмм. Для нагрузочного тестирования использовалась библиотека Locust [11]. От основного микросервиса требовалась средняя задержка отклика не более 500 мс, а от других микросервисов – не более 1000 мс при нагрузке в 100 пользователей. Результаты нагрузочного тестирования основного микросервиса отражены в таблицах 3 и 4.

Таблица 3

Статистика запросов к основному микросервису

Маршрут	Запросы	Ошибки	Среднее (мс)	Мин. (мс)	Макс. (мс)	Сред. размер (байт)	RPS	Ошибки / с
/levels	6743	0	85	3	676	506	111.8	0.0
/stats	20486	0	161	2	1151	331	339.5	0.0
Итого	27229	0	142	2	1151	374	451.3	0.0

Таблица 4

Статистика ответов основного микросервиса

Маршрут	50%ile (мс)	60%ile (мс)	70%ile (мс)	80%ile (мс)	90%ile (мс)	95%ile (мс)	99%ile (мс)	100%ile (мс)
/levels	78	94	110	130	160	180	220	680
/stats	110	140	190	270	370	460	630	1200
Итого	100	130	160	210	330	430	600	1200

Показатели, полученные в ходе нагрузочного тестирования, находятся в допустимых пределах, тестирование прошло успешно.

Заключение. В рамках представленной статьи был рассмотрен процесс проектирования и тестирования программной подсистемы тестирования знаний языков описания аппаратуры. Были рассмотрены варианты использования подсистемы, ее архитектура, симуляция работы цифровых устройств и преобразование временных диаграмм в ее рамках. Также, было проведено функциональное и нагрузочное тестирование подсистемы, все тесты завершились успешно.

Литература

[1] Е.А. Ильина, Л.Г.Егорова, А.В. Дьяконов. Технология тестирования знаний студентов с использованием системы Moodle. Математическое и программное обеспечение систем в промышленной и социальной сферах. Магнитогорск, ГОУ ВПО "Магнитогорский государственный технический университет им. Г.И. Носова", 2011.

[2] Иванова Г.С. Технология программирования. М., Кнорус, 2016.

[3] C4 model. *c4model.com: веб-сайт.* URL: https://c4model.com/ (дата обращения: 01.02.2023).

[4] С.И. Мычко. Микросервисная архитектура. Рязань, Рязанский государственный радиотехнический университет, 2019.

[5] Основы правил проектирования базы данных. *habr.com: веб-сайт.* URL: https://habr.com/ru/articles/514364/ (дата обращения: 05.03.2023).

[6] Симуляция проекта с помощью Icarus-Verilog. *marsohod.org: веб-сайт.* URL: https://marsohod.org/11-blog/113-icarus (дата обращения: 13.03.2023).

[7] PyDigitalWaveTools. *github.com: веб-сайт.* URL: https://github.com/Nic30/pyDigitalWaveTools (дата обращения: 27.03.2023).

[8] Hitchhiker's Guide to the WaveDrom. *wavedrom.com: веб-сайт.* URL: https://wavedrom.com/tutorial.html (дата обращения: 03.04.2023).

[9] Зиганшина М.Р., Валиуллина Д.И., Зиганшин И.А. Применение фреймворка Pytest для тестирования программного кода на языке Python. Казань, ФГБОУ ВО "Казанский национальный исследовательский технологический университет", 2022.

[10] Allure — фреймворк от Яндекса для создания отчётов автотестов. *habr.com: веб-сайт.* URL: https://habr.com/ru/companies/yandex/articles/232697/ (дата обращения: 12.04.2023).

[11] Locust - A modern load testing framework. locust.io: *веб-сайт.* URL: https://locust.io/ (дата обращения: 18.04.2023).