Практическое применение методологии GitOps и CICD подходов при разработке систем на ПЛИС

П.С. Цынгалёв, А.С. Боронников, Г.М. Кряхтунов

***Аннотация* — В настоящее время в России наблюдается рост использования программируемых логических интегральных схем (ПЛИС) благодаря их гибкости, универсальности и возможности многократного перепрограммирования. Однако разработка проектов на ПЛИС сопряжена с трудностями, связанными с ресурсоемкостью используемых систем автоматизированного проектирования (САПР) и отсутствием единых подходов к автоматизации процессов разработки и сборки проектов.**

Данная работа посвящена исследованию применения методологии GitOps и подходов CICD в области разработки проектов на ПЛИС. Проведен анализ их потенциальных преимуществ и недостатков, рассмотрено использование семантического версионирования для управления проектами. Разработаны и предложены два варианта CICD-конвейеров, а также сформирован перечень ключевых этапов автоматизации и необходимых компонентов инфраструктуры платформы для реализации применения GitOps и CICD.

Результаты работы могут быть полезны для оптимизации процессов разработки аппаратных устройств, а также для создания конвейеров сборки и тестирования проектов на базе ПЛИС. В дальнейшем планируется провести тестирование предложенного подхода и анализ инструментов для его реализации.

***Ключевые слова* — DevOps, CICD, конвейеры, информационная система, инфраструктура, автоматизация, оптимизация, ПЛИС.**

I. Введение

В современных условиях развития технологий в России наблюдается значительный рост числа проектов, связанных с использованием программируемых логических интегральных схем (ПЛИС). Это связано с их универсальностью, высокой гибкостью и возможностью многократного перепрограммирования, что делает ПЛИС идеальной платформой для создания прототипов и отладки микропроцессорных систем перед их массовым производством или разработке отдельных функциональных блоков [2, 3, 4].

Основными языками описания аппаратуры в данной области являются Verilog, SystemVerilog и VHDL, а инструментами – системы автоматизированного проектирования (САПР) такие, как Xilinx Vivado и Quartus, получившие в нашей стране и мире большую популярность. Эти инструменты предоставляют разработчикам широкий спектр возможностей для описания, моделирования и синтеза сложных цифровых систем. Однако, их использование сопряжено с определенными трудностями. Например, САПР Xilinx Vivado или Quartus отличаются высокой ресурсоемкостью и выполняют множество вложенных операций внутри себя, которые могут быть необязательные на конкретном этапе разработки и скорость выполнения зависит от конфигурации конкретного оборудования, на котором установлена система автоматизированного проектирования, что может значительно замедлить процесс сборки и тестирования проектов.

Применение методологии GitOps и подходов CICD потенциально позволит ускорить процесс разработки проекта в рамках команды разработчиков за счет оптимизации и автоматизации рутинных задач. Также анализ данной области показал, что нет общего подхода к процессам автоматизации разработки и сборки проектов, большинство решений являются локальными для каждой компании и не освещены [8, 9].

Таким образом, данная работа посвящена исследованию применимости GitOps и CICD подходов при переносе в контекст разработки и сборки проектов на ПЛИС.

II. GitOps и CICD подходы

GitOps – это методология, которая использует Git в качестве основного источника информации для управления инфраструктурой системы и приложениями. Она позволяет описывать желаемое состояние системы декларативно в коде, а средства автоматизации применяют эти изменения к реальной инфраструктуре платформы. GitOps обеспечивает непрерывное согласование между желаемым и фактическим состоянием системы, что обеспечивает согласованность и воспроизводимость конфигурации [1, 9].

Ключевые моменты GitOps:

– описание инфраструктуры носит декларативный характер: описывается в файлах в различных

Статья получена 12 марта 2025.

Павел Сергеевич Цынгалёв, Кафедра вычислительной техники, МИРЭА – Российский технологический университет, Москва, Россия (e-mail: pstsyngalev@mail.ru).

Антон Сергеевич Боронников, Кафедра вычислительной техники, МИРЭА – Российский технологический университет, Москва, Россия (e-mail: boronnikov-anton@mail.ru).

Глеб Михайлович Кряхтунов, Кафедра вычислительной техники, МИРЭА – Российский технологический университет, Москва, Россия (e-mail: gleb.kryakhtunov@yandex.ru).

форматах, например, YAML, JSON, Puppet DSL, для описания желаемого состояния инфраструктуры платформы.

– использование Git: репозиторий Git служит основным местом хранения и управления конфигурациями, ресурсами программного обеспечения.

– автоматическое согласование: инструменты постоянно сравнивают фактическое состояние с желаемым и применяют изменения.

CICD (Continuous Integration/Continuous Delivery) — это подходы, которые позволяют автоматизировать процесс разработки и развертывания программного обеспечения. CI включает в себя непрерывную интеграцию кода, его сборку и тестирование. CD обеспечивает непрерывную поставку готового продукта в производственную среду [1, 9]. GitOps можно рассматривать как расширение CICD, которое применяется не только для кода приложения, но и для архитектуры информационной системы.

Ключевые принципы CICD:

– непрерывная интеграция (CI): автоматическое тестирование и сборка кода после каждого изменения.

– непрерывная поставка (CD): автоматическое развертывание готового продукта в производственную среду.

Связь между методологией GitOps и CICD-подходами заключается в том, что GitOps использует инструменты CICD для автоматизации развертывания и обновления инфраструктуры системы на основе изменений в репозитории Git. Таким образом, GitOps может быть рассмотрен как расширение CICD, которое применяет эти принципы не только к коду приложений, но и к инфраструктуре платформы.

Преимущества совместного использования GitOps и CICD:

– повышение производительности: автоматизация процессов позволяет разработчикам сосредоточиться на написании кода.

– повышенная безопасность: контроль доступа и аудит через Git.

– быстрое развертывание: автоматизация ускоряет процесс сборки, тестирования и развертывания изменений.

В целом, GitOps и CICD дополняют друг друга, позволяя создавать эффективные и масштабируемые рабочие процессы для управления как программными приложениями, так и инфраструктурой систем [6].

III. Проблемы при разработке и сборке проектов на базе ПЛИС

Традиционный маршрут разработки проектов на ПЛИС (рис. 1) включает последовательную реализацию модулей, затем проходит статический анализ (линтинг), результатом которого является RTL-модель (Register Transfer Level), синтез (раскладывание RTL схемы на примитивы ПЛИС), имплементация (трассировка компонентов по кристаллу) и генерация файла конфигурации ПЛИС (bit-файл). Параллельно с линтингом и синтезом можно проводить верификацию через программную симуляцию, при этом каждый модуль зачастую проверяется по очереди [2-5].

Из-за последовательных действий и выполнения каждого этапа на персональной машине сложно распараллелить процесс разработки, разработчикам затруднительно одновременно создавать и отлаживать несколько новых модулей или проектов, так как верификация и синтез требуют значительного времени. Если в команде есть общий сервер с установленным САПР, то его можно использовать для распараллеливания процесса синтеза и сборки проектов, но при этом необходимо вручную заходить в удаленную систему, запускать процессы и ожидать своей очереди, что существенно замедляет процесс разработки. Кроме того, отладка некоторых функциональных блоков может занимать значительное время, дополнительно блокируя процесс разработки и снижая общую эффективность работы [2, 3].

Большинство САПР имеет множество дополнительных настроек по умолчанию, которые не всегда нужны на конкретном этапе проектирования устройства. Для избежание таких случаев можно вручную конфигурировать САПР с помощью TCL-скриптов. Написание таких пользовательских сценариев достаточно ресурсоемкая задача и не всегда реализуемая.

Зачастую САПР распространяются через платную лицензию и требует дополнительных компетенций от разработчика, что является существенными недостатками при проектировании систем под разные семейства ПЛИС.

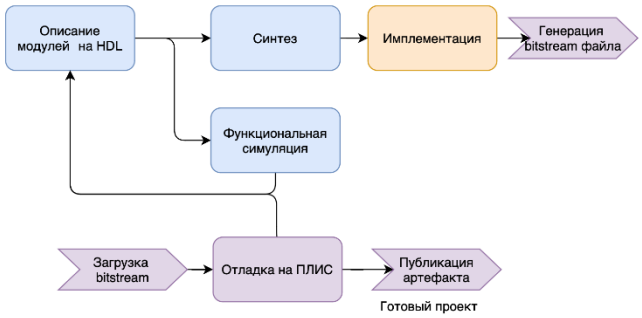

Рисунок 1 – Традиционный маршрут проектирования

Таким образом, применимость подходов GitOps и CICD, описанных в Разделе 2 данной статьи, позволяет получить следующие преимущества, по сравнению с традиционной разработкой проектов [6-9]:

GitOps:

– упрощение управления инфраструктурой системы: GitOps методология подразумевает использование единого места хранения исходного кода проекта, в которое разработчики постепенно добавляют новые функциональные блоки.

– повышение прозрачности и контроля: все изменения инфраструктуры версионируется через Git, что позволяет легко отслеживать историю изменений и возвращаться к предыдущим версиям проекта в случае ошибок.

– автоматизация развертывания: благодаря GitOps можно описать конвейеры, которые будут учитывать

каждый этап разработки, сборки и тестирования проекта.

– увеличение скорости разработки: использование Git для управления проектов позволит распараллелить работу нескольких сотрудников или команд над одним проектом, поскольку все изменения проходят через изолированные ветви с конвейерами.

– масштабируемость и надежность: GitOps обеспечивает гибкую воспроизводимость конфигураций, что позволит их распространять и тонко настраивать под нужды различных проектов.

– мнижение затрат и рисков: автоматизация и отслеживание изменений позволяют быстрее выявлять и исправлять ошибки, что снижает затраты на поддержку и минимизирует риск критических сбоев.

CICD:

– сокращение сроков разработки: CICD позволяет сократить время между внесением изменений и их развертыванием, что ускоряет вывод продуктов на рынок.

– повышение качества кода: автоматические тесты на этапе непрерывной интеграции помогают выявлять ошибки на ранних стадиях, что улучшает общее качество кода и снижает вероятность критических ошибок.

– автоматизация развертывания: благодаря CICD можно реализовать конвейеры, которые будут учитывать каждый этап разработки, сборки и тестирования проекта.

– повышение надежности и стабильности: CICD обеспечивает автоматическое тестирование и развертывание, что снижает риск ошибок и повышает стабильность программного обеспечения.

– экономия ресурсов: автоматизация процессов снижает нагрузку на разработчиков и позволяет сосредоточиться на стратегических задачах, а не на ручной сборке и тестировании.

– улучшение опыта пользователей (разработчиков): быстрые и регулярные обновления позволяют оперативно добавлять новую функциональность и исправлять ошибки, что повышает удовлетворенность пользователей.

Но при этом есть и ряд достаточно весомых недостатков применения методологии и подхода:

– повышение сложности и стоимости инфраструктуры для проведения разработки, тестирования и сборки проектов;

– повышение стоимости разработки за счет ввода новых компетенций;

– затраты на проектирование процессов CICD конвейеров и их внедрения.

Несмотря на выявленные недостатки, применение методологии GitOps и подходов CICD является целесообразной не только в контексте разработки программных продуктов, но также потенциально позволяет повысить скорость и качество разрабатываемых устройств на базе ПЛИС [5-7].

IV. Применение GitOps и CICD подходов к решению проблем при разработке и сборке проектов проектов на базе ПЛИС

Процесс разработки устройств на ПЛИС, исходя из Раздела 3, можно разделить на несколько ключевых этапов (стадий), которые могут включать отдельные инструкции и могут быть автоматизированы при помощи CICD конвейеров:

1. Линтинг, синтез и верификация модуля с помощью симуляций: процесс проверки и преобразования исходных файлов проекта, которые написаны, например, на языке VHDL или Verilog, вначале в RTL-модель, после разложение на логические примитивы, готовые для имплементации на ПЛИС. Параллельно с построением RTL-модели и синтезом выполняется проверка корректности работы модуля через симуляцию его работы в среде моделирования. На данном этапе подключаются различные симуляторы.

2. Сборка (имплементация) bit-файла: результатом имплементация является создание bit-файла, который загружается в память для конфигурирования ПЛИС.

3. Верификация модуля на ПЛИС: на этом этапе выполняется загрузка bit-файла в конфигурационную память ПЛИС для проверки работы модуля в реальной аппаратной среде.

4. Публикация артефакта (в данном случае артефактом является полученный bitstream файл) в репозитории: после успешной сборки и тестирования, новая версия проекта или артефакт (например, bit-файл) публикуется в репозитории, где он может быть использован другими разработчиками или командами.

Каждый этап при этом выполняется в определенном сегменте инфраструктуры и должен иметь гибкую конфигурацию каждой стадии. Это достигается как раз за счет применения GitOps методологии [2-7, 11].

При этом возникает проблема версионирования с ростом количества модулей и развитием проекта. Для ее решения можно использовать семантическое версионирование из разработки программного обеспечения.

Такая система имеет следующий формат: версии имеют вид “<x>.<y>.<z>-<prefix>”, где x – мажорная версия, y – минорная версия, z – патч-версия [12]. При этом:

– мажорная-версия повышается, когда сделаны обратно несовместимые изменения;

– минорная-версия повышается, когда добавляется новая функциональность, не нарушая обратной совместимости;

– патч-версия повышается, когда происходят обратно совместимые исправления.

Общая схема публикации изменений разработчиком и примером семантического версионирования представлена на рис. 2.

Каждый разработчик ведет работу над своей частью в отдельной ветке “feature” которая потом сливается с основной ветки командной разработки “development”, которых может быть также несколько, при завершении разработки формируется релиз. Для каждого изменения запускаются конвейеры в зависимости от целевой ветки и правил, наложенных на нее.

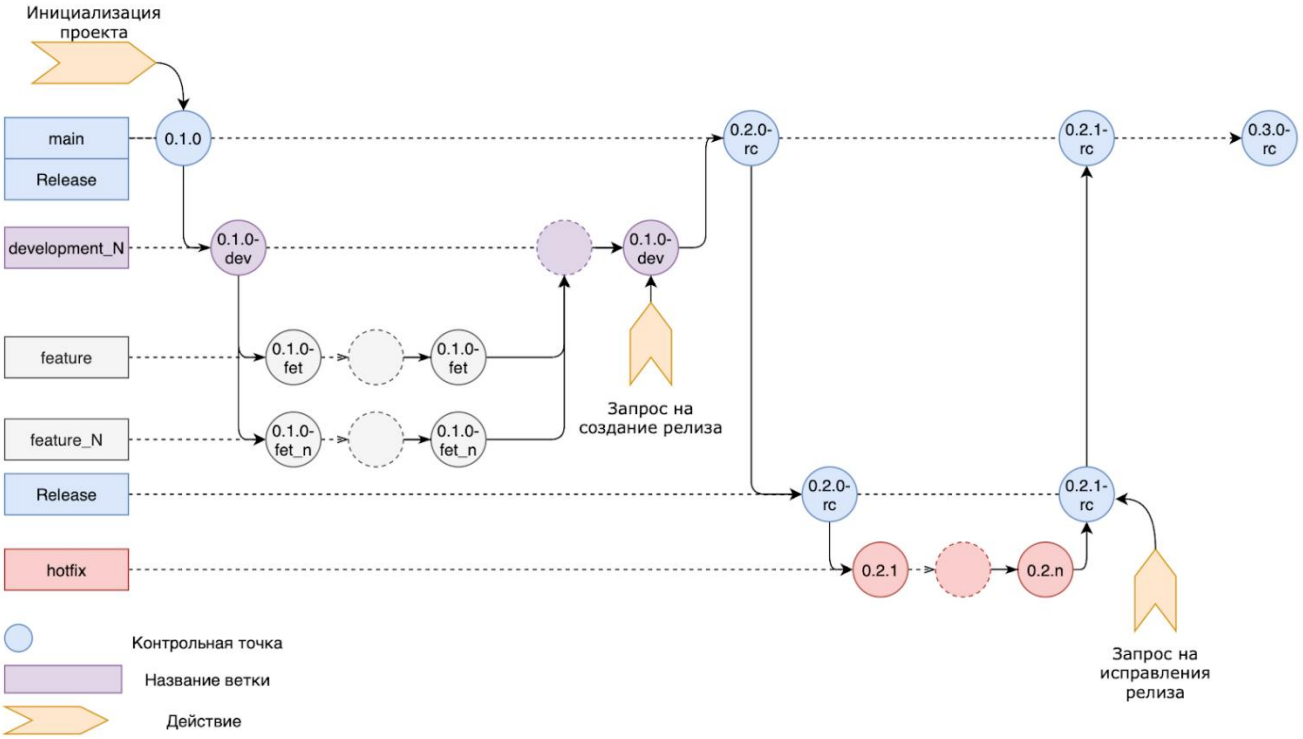

Рисунок 2 – Схема ключевых действий при версионировании проекта

Далее при проектировании CICD конвейеров, появляется проблемы конфигурирования проекта для запуска каждого из этапов. Основные ключевые проблемы:

– выбор семейства и модели ПЛИС;
– выбор симулятора;
– выбор структуры проекта (зависит САПР);
– выбор файла проектных ограничений;
– выбор модулей верхнего уровня;
– определение зависимостей модулей.

Для решения этих проблем можно предусмотреть отдельные надстройки в конфигурационном файле для CICD конвейеров, в зависимости от которых будет выполняться сборка проекта под разную структуру, зависящую, например, от различных САПР, в которых ведется разработка с учетом семантического версионирования. Общая схема конвейера для нескольких САПР представлена на рис. 3-4. Для удобства проведения тестирования и оптимизации использования ресурсов инструкция по тестированию на ПЛИС для релизной ветки запускается автоматически, а для остальных – в ручном режиме.

```
...
CAD: #<тип САПР>
CONSTRAINS: #<файл проектных ограничений>
TOP_MODULE: #<модуль верхнего уровня>
SIMULATOR: #<тип симулятора>
...
```

Рисунок 3 – Пример структуры конфигурационного файла с учетом среды разработки проекта

При этом такая схема не является универсальной и при смене ПЛИС, отладочной платы, конфигурации проекта нужно редактировать конфигурационный файл конвейера. Также необходимо разработать отдельные инструкции для обработки иерархий хранения конфигурационных файлов под каждый тип САПР и получение остальной конфигурации, которая не определена в конфигурационном файле конвейера, но необходима для сборки и тестирования проекта, что является достаточно ресурсоемкой задачей.

Другой способ – унифицировать описание конфигурации проекта перед сборкой, можно реализовать отдельный сервис и передавать ему созданный конфигурационный файл (несколько файлов для различных сред тестирования) определенного формата (например YAML) – рис. 5 (пример конфигурации) и рис. 6, при этом усложняется процесс конфигурации проекта и требуется дополнительные ресурсы на реализацию и поддержку такого сервиса, но отпадает необходимость при использовании новых САПР, смены чипа ПЛИС реконфигурировать структуру проекта и настройки CICD-конвейеров, а также реализовать новые конвейеры для новых САПР.

Данный сервис формирует инструкции для запуска каждого из этапов CICD конвейера и позволяет гибко конфигурировать настройки проекта. Тестирование на ПЛИС запускается в ручном режиме (для веток разработки, при релизе используется принудительный режим) поскольку могут возникнуть исключительные ситуации, когда разработчику необходимо вручную править RTL модель перед генерацией bit-файла и загрузки его на ПЛИС.

Таким образом, все это позволит значительно ускорить процесс разработки проекта за счет автоматизации рутинных задач, и как результат повысить эффективность распределения ресурсов команд участвующих в разработке проектов организации.

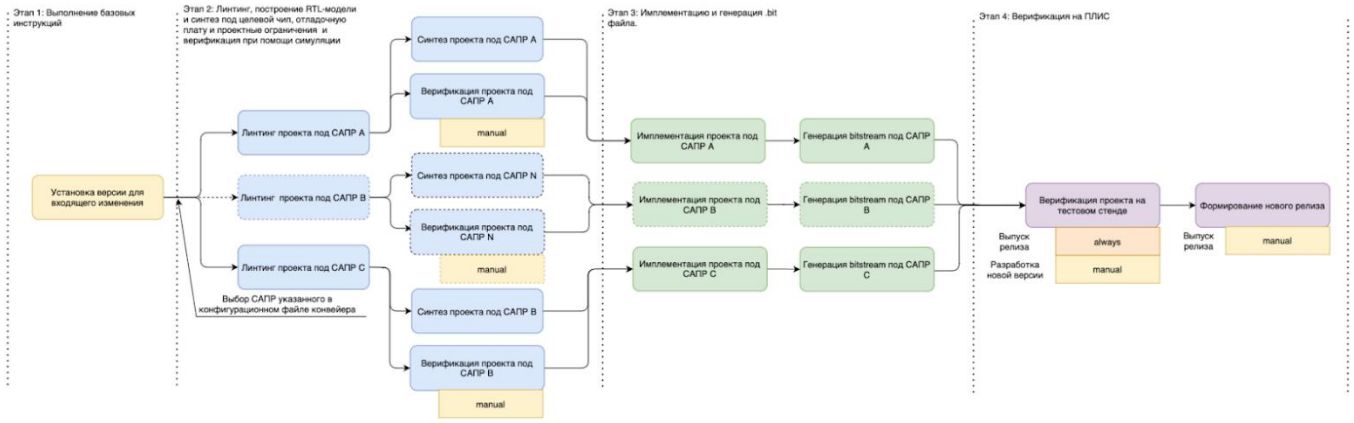

Рисунок 4 – Этапы CICD конвейеров для каждой ветки с учетом среды разработки проекта

```yaml
fpga_package:
  - name: #<имя проекта>
    cad: #<тип САПР>
    crystal:
      - #<тип кристалла>
    project_settings_file_path: /path/to/file
    constrains:
      - name: testA
        path: /path/to/file
        group: A
      - name: testB
        path: /path/to/file
        group: A
      - name: testC
        path: /path/to/file
        group: B
    top_modules:
      - name: topA
        path: /path/to/file
      - name: topB
        path: /path/to/file
    test_bench_relations:
      - module_name: #<имя модуля>
        path_module: path-to-file
        test_benchs:
            path_test_bench:
              - #<имя файла симуляции №1 текущего модуля>
              - #<имя файла симуляции №N текущего модуля>
    synthes_scheme: path-to-file
    synthes_settings:
      enable_warngings: true
      skip_rtl: true
    synthes_settings:
      simulator_name: #<имя симулятора>
    dependences: # Пути до необходимых модулей, можно указать директоию через /path/to/dir/*
      - path: /path/to/file
```

Рисунок 5 – Пример структуры использовании универсального конфигурационного файла

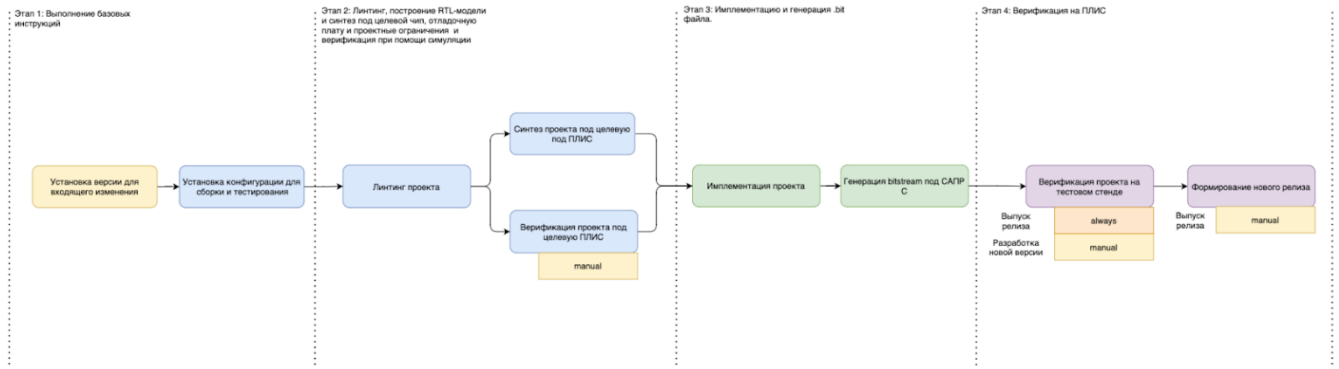

Рисунок 6 – Этапы CICD конвейеров для каждой ветки при использовании универсального конфигурационного файла

V. Архитектура программной системы разработки и сборки

Методология GitOps позволяет описать гибко-конфигурируемую инфраструктуру для реализации ранее описанных CICD конвейеров. При этом архитектура такой системы должна включать следующие компоненты:

– система контроля версии;

– система хранения и управления репозиториям систем контроля версии – выполнять роль такой системы

могут решения с открытой лицензией, например Gitlab, GitHub, BitBucket;

– система сборки проектов под ПЛИС – при этом если используется универсальный способ конфигурирования проекта данная система должна включать сервис для обработки файла универсальной конфигурации проекта и последующего вызова инструкций;

– среда для построения CICD конвейеров – может быть встроена в систему хранения и управления репозиториям систем контроля версии;

– среда исполнения процессов сборки проектов – это может быть среда ОС, виртуализированная среда, или контейнеризированная среда;

– среда разработки проектов – среда, в которой пользователь выполняет основную разработку проекта (Xilinx Vivado, Xilinx ISE 14.7, Quartus и т.д);

– среда хранения артефактов сборки – единый репозиторий для хранения версий bit-файлов для программирования ПЛИС;

– программное обеспечение для подключения к отладочным платам с целевыми микросхемы ПЛИС для проведения “физического” тестирования – может быть реализовано как отдельный сервис в системе, который выполняет роль управления над ПЛИС, собранных в кластер;

– стенд с подключенными ПЛИС для проведения тестирования – при использовании облачной инфраструктуры для сборки и тестирования необходимо обеспечить канал для доступа к ПЛИС.

Схема архитектуры описанной системы показана на рис. 7. Сервера для сборки системы могут находится непосредственно в центре проектирования с разработчиками так и в облачной среде. Сами ПЛИС при этом должны быть подключены либо к выделенному серверу, который связан с облаком или выделенной средой, например, через vpn-канал, либо напрямую к серверам, если используется локальная инсталляция, либо через некоторый usb-хаб который подключен к облаку разработки. При этом также рекомендуется организовывать связь через vpn-канал между рабочими местами разработчиков и платформой сборки и тестирования в случае использования облачной или выделенной среды [2, 3, 6, 7, 11].

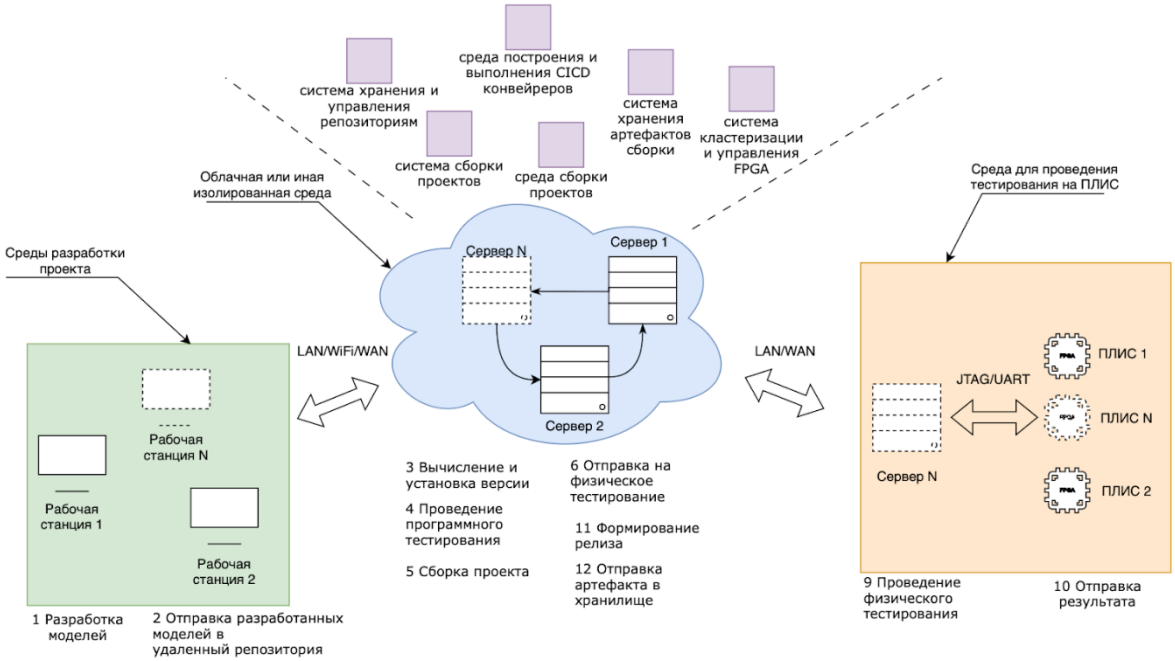

Рисунок 7 – Архитектура распределенной системы для автоматизированной разработки и сборки проектов на ПЛИС

VI. Заключение

В результате проведения работы была исследована возможность применения методологии GitOps и подходов CICD из области разработки программного обеспечения к области разработки аппаратных устройств. Были выявлены преимущества и недостатки применение методологии и подхода в новой парадигме. Предложено применение семантического версионирования к разработке проектов на базе ПЛИС. Разработана схема ведения проекта с применением семантического версионирования.

Также были выявлены недостатки при традиционном подходе разработки проектов на базе ПЛИС. На основе этого были выделены ключевые этапы для автоматизации и описаны необходимые действия при проектировании CICD конвейеров. Выдвинута гипотеза об эффективности применения синергии методологии GitOps и подхода CICD.

В ходе выполнения работы разработаны и предложены 2 способа построения CICD конвейеров с указанием их преимуществ и недостатков. Сформированный список необходимых компонентов для построения инфраструктуры системы автоматизированной разработки, сборки и тестирования проектов. Предложен вариант архитектуры системы.

В дальнейших исследованиях планируется провести анализ инструментов, позволяющих выстраивать инфраструктуру согласно предложенной архитектуре.

Также провести тестирования для практического подтверждения эффективности применения описанного подхода.

Библиография

[1] Ермаков А.С. Перспективное развитие методологии DevOps // Вестник НГУЭУ, 2020 – С. 174-183.

[2] Эннс В.И., Гаврилов С.В., Чочаев Р.Ж. Автоматическая настройка программных средств размещения пользовательских схем на плис // Известия высших учебных заведений. Электроника, 2021 – Т. 26, № 6 – С. 508-520.

[3] Еськов В.С. Исследование способов реинжиниринга загрузочной последовательности ПЛИС // Инновации в информационно-аналитических системах, 2014 – №. 2, С. 89-94.

[4] Нужнов Е.В., Полупанов А.А. Особенности и возможности автоматизированного проектирования ПЛИС различной архитектуры // Известия Южного федерального университета. Технические науки, 2008 – Т. 81, № 4 – С. 55-61.

[5] Брехов О.М., Ратников М.О. Сравнительный анализ тестовых систем плис и их окружения // Труды МАИ, 2022 – № 125 – С. 529-577.

[6] Brennan Wilkes Alessandra Maciel Paz Milani Margaret-Anne Storey "A Framework for Automating the Measurement of DevOps Research and Assessment (DORA) Metrics" 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME), Year: 2023, Pages: 62-72 DOI Bookmark: 10.1109/ICSME58846.2023.00018

[7] Jean-Didier Totow Tom-Ata, Kyriakos Kritikos, Maria Antoneta Di Girolamo, Efterpi Paraskevoulakou, Chrysostomos Symvoulidis, Dimosthenis Kyriazis "Polymorphic Cloud Application Design Assisted by Open Source Software Classification", 2023 5th International Communication Engineering and Cloud Computing Conference (CECCC) 27-29.09;2023 DOI: 10.1109/CECCC59577.2023.10560739

[8] Nerina Peña Olivero, Himer Ávila George, Gabriel Alberto García-Mireles "Impact of Devops Practices on Software Product Quality: Preliminary Findings From a Systematic Mapping", 2023 12th International Conference On Software Process Improvement (CIMPS) 18-20 October 2023 DOI 10.1109/CIMPS61323.2023.10528820

[9] Saumya Gupta, Madhulika Bhatia, Meenakshi Memoria, Preeti Manani "Prevalence of GitOps, DevOps in Fast CI/CD Cycles", 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON) 18-20 October 2023 DOI: 10.1109/CIMPS61323.2023.10528820

[10] Игошина М.С. Особенности сред моделирования плис // Теория и практика современной науки, 2021 – № 11 (77) – С. 224-227.

[11] Опыт автоматизации управления FPGA-стендами для распределенной команды. – URL: https://engineer.yadro.com/article/fpga-remote/

[12] Семантическое версионирование. – URL: https://semver.org/lang/ru/

Practical application of GitOps methodology and CICD approaches in the development of FPGA systems

P.S. Tsyngalev, A.S. Boronnikov, G.M. Kryakhtunov

***Abstract* — Currently, there is an increase in the use of programmable logic integrated circuits (FPGAs) in Russia due to their flexibility, versatility and the possibility of multiple reprogramming. However, the development of FPGA projects is fraught with difficulties related to the resource intensity of computer-aided design (CAD) systems used and the lack of unified approaches to automating project development and assembly processes.**

This paper is devoted to the study of the application of GitOps methodology and CICD approaches in the field of FPGA project development. The analysis of their potential advantages and disadvantages is carried out, the use of semantic versioning for project management is considered. Two variants of CICD pipelines have been developed and proposed, as well as a list of key automation stages and necessary components of the platform infrastructure for implementing GitOps and CICD applications.

The results of the work can be useful for optimizing hardware device development processes, as well as for creating assembly pipelines and testing FPGA-based projects. In the future, it is planned to test the proposed approach and analyze the tools for its implementation.

***Keywords* — DevOps, CICD, pipelines, information system, infrastructure, automation, optimization, FPGA.**

REFERENCES

[1] Ermakov A.C. Promising development of DevOps methodology // Bulletin of the National University of Economics, 2020 – pp. 174-183.

[2] Enns V.I., Gavrilov S.V., Chochaev R.J. "Automatic configuration of software tools for placing user circuits on FPGAs" Izvestia of higher educational institutions. Electronics, vol. 26, No. 6, 2021, pp. 508-520.

[3] Eskov V.S. "Research of ways of reengineering the FPGA boot sequence" Innovations in information and analytical systems, No. 2, 2014, pp. 89-94

[4] Nuzhnov E.V., Polupanov A.A. "Features and possibilities of computer-aided design of FPGAs of various architectures" Izvestiya Yuzhnogo federalnogo universiteta. Technical Sciences, vol. 81, No. 4, 2008, pp. 55-61.

[5] Brekhov O.M., Ratnikov M.O. "Comparative analysis of FPGA test systems and their environment" Proceedings of MAI, No. 125, 2022, pp. 529-577.

[6] Brennan Wilkes Alessandra Maciel Paz Milani Margaret-Anne Storey "A Framework for Automating the Measurement of DevOps Research and Assessment (DORA) Metrics" 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME), Year: 2023, Pages: 62-72 DOI Bookmark: 10.1109/ICSME58846.2023.00018

[7] Jean-Didier Totow Tom-Ata, Kyriakos Kritikos, Maria Antoneta Di Girolamo, Efterpi Paraskevoulakou, Chrysostomos Symvoulidis, Dimosthenis Kyriazis "Polymorphic Cloud Application Design Assisted by Open Source Software Classification", 2023 5th International Communication Engineering and Cloud Computing Conference (CECCC) 27-29.09;2023 DOI: 10.1109/CECCC59577.2023.10560739.

[8] Nerina Peña Olivero, Himer Ávila George, Gabriel Alberto García-Mireles "Impact of Devops Practices on Software Product Quality: Preliminary Findings From a Systematic Mapping", 2023 12th International Conference On Software Process Improvement (CIMPS) 18-20 October 2023 DOI 10.1109/CIMPS61323.2023.10528820

[9] Saumya Gupta, Madhulika Bhatia, Meenakshi Memoria, Preeti Manani "Prevalence of GitOps, DevOps in Fast CI/CD Cycles", 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON) 18-20 October 2023 DOI: 10.1109/CIMPS61323.2023.10528820

[10] Igoshina M.S. "Features of FPGA modeling environments" Theory and Practice of modern Science, No. 11 (77), 2021, pp. 224-227.

[11] Experience in automation of FPGA control stands for a distributed team. – URL: https://engineer.yadro.com/article/fpga-remote/

[12] Semantic versioning. – URL: https://semver.org/lang/ru/