ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 11**, № 1(44), с. 31–55

ББК 33971.35:3972.53
ГРНТИ 50.33.04, 50.41.17
УДК 004.32:519.691

И. А. Адамович, Ю. А. Климов

# Пакетный протокол взаимодействия программируемых логических интегральных схем

Аннотация. При создании вычислительных систем с использованием программируемых логических интегральных схем (ПЛИС) или специализированных микросхем часто встает проблема соединения нескольких микросхем между собой для передачи данных. При создании авторами подобной вычислительной системы оказалось, что существующие доступные протоколы взаимодействия не обладают требуемыми свойствами: пакетной передачей, использованием небольшого числа линий ввода-вывода общего назначения, необходимой пропускной способностью.

Представлен пакетный протокол передачи данных между ПЛИС, использующий 6 линий ввода-вывода общего назначения и обеспечивающий скорость передачи до 10 МБ/с (суммарно в обе стороны) при частоте 20 МГц.

*Ключевые слова и фразы:* полудуплексная передача, кредитная информация, сериализация/десериализация данных, конечный автомат, сдвиговый регистр, язык описания аппаратуры.

## Введение

При создании различных вычислительных систем с использованием программируемых логических интегральных схем (ПЛИС) или специализированных микросхем часто встает проблема соединения нескольких микросхем между собой для передачи данных.

В некоторых случаях возможно применение широко используемых интерфейсов таких как I2C или UART, которые, однако, обладают низкой скоростью передачи данных. Существенными достоинствами этих интерфейсов является использование только двух линий ввода-вывода для передачи данных, а также их широкое распространение: аппаратная реализация интерфейсов I2C или UART присутствует во многих микроконтроллерах и микропроцессорах. Существуют открытые реализации этих интерфейсов для использования их в ПЛИС.

Работа Ю.А. Климова выполнена при поддержке гранта РНФ, проект № 19-71-30004.



Поэтому, когда необходимо соединить ПЛИС и микроконтроллер или микропроцессор, часто используются эти интерфейсы.

При необходимости обеспечить высокую скорость передачи данных можно использовать высокоскоростные интерфейсы, которые используют либо большое число линий ввода-вывода, либо специальные дифференциальные линии.

Шины с большим числом линий ввода-вывода активно использовались ранее (например, шины ISA и PCI), когда частота передачи данных была близка к частоте работы микросхем. Сейчас область их применения ограничена из-за широкого использования дифференциальных линий. Основные недостатки таких шин происходят из-за необходимости жесткой синхронизации большого числа линий: имеются ограничения на длину линий, сложности разводки плат с большим числом линий и, как следствие, ограниченность частоты передачи данных.

В настоящее время там, где требуется очень высокая скорость (от 1 ГБ/с и выше) и/или необходимо передавать данные на большое расстояние, широко используются дифференциальные линии (например, PCI-Express, Ethernet или USB). Дифференциальные линии и, во многих случаях, аппаратные блоки, реализующие широко распространенные пакетные интерфейсы, встроены во многие ПЛИС. Например, для работы с сетью Ethernet фирма Intel Corp. предоставляет для своих ПЛИС блок Triple-Speed Ethernet IP Core[1], а Xilinx Inc. — блок Tri-Mode Ethernet MAC[2]. Так же доступны сторонние блоки для работы с сетью Ethernet на уровне UDP, например, ядро e7 UDP / IP[3] от e-trees.Japan Inc. Последний использовался в работе [**1**] для реализации протокола CoAP для интернета вещей. В некоторых случаях на основе встроенных аппаратных блоков разрабатывают собственные пакетные протоколы, как, например, в работе [**2**].

В целом их применение оправдано, когда требуется соединить небольшое число ПЛИС с высокой скоростью передачи (обычно в ПЛИС встраивается 1–4 блока Ethernet или PCI-Express, что позволяет к одной микросхеме подключить только 1–4 других) или подключить ПЛИС к процессору, поддерживающему подобный интерфейс. В случае

[1]Triple-Speed Ethernet IP Core, URL https://www.intel.com/content/dam/www/program/design/us/en/documents/low-pin-count-interface-specification.pdf

[2]Tri-Mode Ethernet Media Access Controller, URL https://www.xilinx.com/products/intellectual-property/temac.html

[3]e7 UDP / IP, URL https://e-trees.jp/e7-udp-ip/

разработки специализированной микросхемы использование дифференциальных линий повышает стоимость микросхемы (требуется лицензирование соответствующих аппаратных блоков) и сложность её разработки, поэтому может не подходить в некоторых случаях.

Между указанными «крайностями» существуют несколько интерфейсов (например, SPI/QSPI и LPC), которые используют невысокое число (4–6) линий ввода-вывода общего назначения (GPIO, general-purpose input/output) и функционируют на частотах нескольких десятков (в отдельных случаях сотен) МГц. Однако шина LPC в настоящее время практически нигде не используется. SPI в своей базовой версии обладает относительно невысокой скоростью. Более высокопроизводительные версии (Quad SPI, QSPI) являются специализированными решениями (в основном, для микросхем флеш-памяти), тянущими за собой груз совместимости. Такие версии нет смысла адаптировать для своих нужд.

В результате анализа существующих интерфейсов было принято решение разработать свой интерфейс для передачи данных между ПЛИС, который был бы симметричным (обе стороны могут независимо друг от друга передавать данные), пакетным (позволило интегрировать данный интерфейс с другими компонентами проекта и минимизировать накладные расходы при передаче больших, до 128 байт в используемой версии, пакетов) и использовать небольшое число (в используемой версии 6) линий ввода-вывода общего назначения.

Дальнейшее изложение построено следующим образом. В разделе 1 приведены требования к разрабатываемому интерфейсу, возникшие в разработанной авторами системе. В разделе 2 приведен обзор существующих распространенных решений с учетом описанных требований. В разделе 3 описан разработанный интерфейс, его физическая и логическая реализации, приведены результаты расчетов и тестирования пропускной способности интерфейса. В заключении подведены итоги разработки.

## 1. Требования к разрабатываемому интерфейсу

Разные решаемые задачи, как и разные физические ограничения вычислительной системы, могут выдвигать различные требования к используемым интерфейсам. Сформулируем требования к интерфейсу, которые выдвинула разрабатываемая авторами вычислительная система.

Главными физическими ограничениями являются доступное количество линий на один порт и требуемая скорость передачи данных.

Если допустимо использовать большое число линий, то возможно применение широких шин с низкой тактовой частотой. Если необходимо обеспечить высокую скорость передачи при небольшом числе линий, то требуется использование высокочастотных дифференциальных линий. Конкретный выбор физического уровня передачи также определяется поддерживаемыми возможностями используемых микросхем.

Разрабатываемая вычислительная система должна состоять из нескольких ПЛИС: одной управляющей ПЛИС и нескольких (до 20) вычислительных ПЛИС, подключенных к управляющей ПЛИС. При разработке в качестве управляющей ПЛИС использовались ПЛИС семейства Xilinx Zynq 7000[4], а в качестве вычислительных ПЛИС использовались ПЛИС семейства Xilinx Artix-7 FPGAs[5]. Основное отличие между указанными семействами, определившее архитектуру системы, заключается в том, что ПЛИС семейства Xilinx Zynq 7000 дополнительно содержат ARM-ядра и периферию (например, DDR-, Ethernet- и USB-контроллеры), позволяющие запустить универсальную ОС Linux на такой микросхеме.

Управляющая ПЛИС Xilinx Zynq 7020 имеет 200 линий ввода-вывода, однако часть линий занята периферией. В результате на один порт можно задействовать не более 8 линий ввода-вывода.

В разрабатываемой системе требовалось обеспечить скорость передачи данных между ПЛИС на уровне 10 МБ/с (суммарно в обе стороны), а так же иметь возможность увеличить скорость передачи до 30–40 МБ/с в будущем. Это приводит к необходимости передавать данные на частоте около 20 МГц (и при 60–80 МГц в будущем), что является подходящей частотой при использовании линий ввода-вывода общего назначения (GPIO, general purpose input-output), поэтому использование дифференциальных линий не требуется. В будущем при необходимости частота передачи может быть повышена до 66–100 МГц.

Основным логическим требованием, вытекающим из специфики задачи, является поддержка симметричной передачи пакетов различной длины (до 128 байт).

Как будет показано в следующем разделе, многие существующие интерфейсы ориентированы на разделение приемника и передатчика на ведущего и ведомого: ведущий может передавать запросы (содержащие адрес и данные), а ведомый выполняет полученные запросы.

---

[4]Xilinx Zynq-7000 SoC, URL https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[5]Xilinx Artix-7 FPGA, URL https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html

В общем случае следующий запрос не может быть отправлен до обработки предыдущего, а также ведомый обычно не может инициировать передачу данных (или требуется дополнительная линия для передачи прерываний).

В создаваемой системе в вычислительных ПЛИС планируется разметить несколько блоков, взаимодействия с которыми должны происходить независимо и неблокирующим образом. Для реализации такого рода задач может применяться пакетная передача, при которой в пакеты упакована вся необходимая информация, а протокол передачи передает пакеты, не понимая что за данные в них находятся.

Использование пакетной передачи позволяет реализовать такие существенные требования, как симметричность и неблокируемость. Симметричность протокола позволяет передавать данные в любую сторону без какой-либо специальной активности другой стороны.

Также важна неблокируемость протокола: возможность временно остановить передачу в одну из сторон, если получатель по какой-либо причине не готов принять новые данные, при этом сохранив способность передавать данные в противоположном направлении.

В заключение отметим, что реализация интерфейса должна быть компактной, чтобы можно было реализовать до 20 портов данного интерфейса в небольшой ПЛИС.

## 2. Обзор существующих интерфейсов

В данной главе будут рассмотрены существующие интерфейсы и, в первую очередь, их протоколы взаимодействия, а также, при необходимости, другие уровни.

### 2.1. I2C

Шина I2C (Inter-Integrated Circuit)[6] — последовательная асимметричная шина, разработанная Philips Semiconductors. Состоит из двух двунаправленных линий связи: линии тактового сигнала (‘SLC’) и последовательной двунаправленной линии данных (‘SDA’). Шина может работать на частоте 100–400 кГц в зависимости от поддерживаемой модификации, что приводит к очень низкой пропускной способности до 50 кБ/с. Из-за невысокой пропускной способности данная шина не может быть использована в разрабатываемой системе.

[6]Inter-integrated circuit (I2C), URL https://en.wikipedia.org/wiki/I2C

Из-за относительной простоты, низкой рабочей частоты и небольшого числа линий многие датчики выпускаются с поддержкой шины I2C. Поэтому большинство микропроцессоров и микроконтроллеров имеют встроенный контроллер шины I2C, что привело к широкому использованию шины I2C для контроля и управления различными микросхемами. Например, в работе [3] рассматривается шина I2C для взаимодействия ПЛИС с датчиком движения (акселерометром и гироскопом).

## 2.2. UART

Интерфейс UART (Universal Asynchronous Receiver/Transmitter)[7], универсальный асинхронный приемопередатчик) — интерфейс для связи цифровых устройств, передающий данные в последовательной форме. Существенными для организации связи являются две линии: линия для передачи и линия для приема (рисунок 1). Важно отметить, что отсутствует линия тактового сигнала, поэтому частота передачи ограничена 100 кГц – 1 МГц.

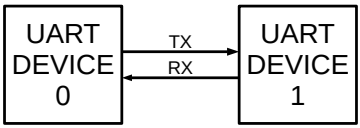


Рисунок 1. Схема интерфейса UART

Опишем базовый цикл передачи слова (рисунок 2). Стандартное слово занимает 8 бит, но интерфейс является настраиваемым и поэтому возможны другие размеры слова. В режиме простоя, когда отсутствует передача, на линии удерживается логическая ‘1’. Начало слова предваряет стартовый бит ‘0’, поэтому приемник UART ждет перехода линии из уровня ‘1’ в уровень ‘0’, от которого отсчитывается прием слова. После стартового бита передаются 8 бит данных, а завершает слово «стоп»-бит, который равен ‘1’.

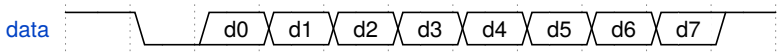


Рисунок 2. Временная диаграмма передачи одного слова в протоколе UART

[7]Universal asynchronous receiver-transmitter (UART), URL https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

UART широко используется для подключения различных датчиков к микропроцессорам и микроконтроллерам, когда пропускной способности I2C недостаточно или требуется двунаправленное взаимодействие. В работе [4] интерфейс UART используется для подключения к датчикам, измеряющим энергопотребление. Каждый датчик имеет два разъема UART, что позволяет их соединить в цепочку. Однако в таком случае время получения данных со всех датчиков будет линейно зависеть от длины цепочки. В этой работе в ПЛИС реализованы несколько интерфейсов UART: по одному для каждого датчика, а также алгоритм параллельного опроса всех датчиков. Что позволило существенно сократить время опроса датчиков.

Из-за недостаточно высокой пропускной способности UART не может быть применен в разрабатываемой системе. Однако отметим существенную для нас особенность интерфейса UART: сигнал начала передачи передается не по отдельной линии (как в некоторых других интерфейсах, описанных ниже), а обозначается изменением уровня сигнала на линии данных. Это позволяет уменьшить количество требуемых линий данных, хотя увеличивает время, требуемое для передачи данных.

## 2.3. SPI

Интерфейс SPI (Serial Peripheral Interface)[8], последовательный периферийный интерфейс) — синхронный стандарт передачи, обеспечивающий простую и недорогую связь микроконтроллеров и периферии. В отличие от UART, SPI является синхронным интерфейсом: для синхронизации ведущего устройства (процессора) с ведомым (периферией) используется линия тактового сигнала, генерируемого ведущим. В результате частота передачи может достигать 33–66 МГц.

К одному ведущему устройству может подключаться несколько ведомых устройств. В таком случае ведущий будет иметь несколько дополнительных выходных линий типа «выбор устройства» (chip select). Периферия, не выбранная таким сигналом, не участвует в обмене по SPI в данный момент времени (рисунок 3).

Передача в SPI осуществляется словами (рисунок 4). Длина слова, как правило, составляет 8 бит. Ведущее устройство инициирует передачу

[8]Serial Peripheral Interface (SPI), URL https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

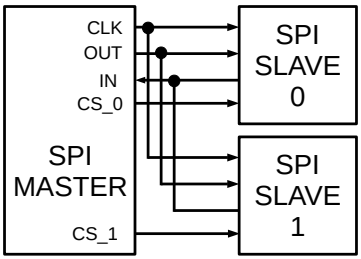


Рисунок 3. Схема интерфейса SPI

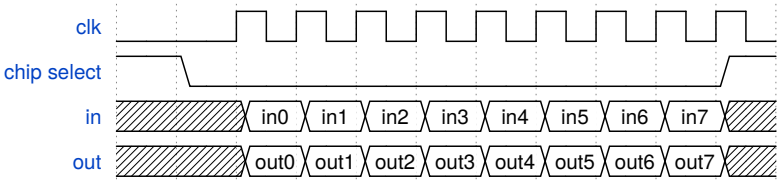


Рисунок 4. Временная диаграмма обмена пакетами между ведущим и ведомым в протоколе SPI

выбором ведомого при помощи линии «выбор устройства». После этого оба, ведущий и ведомый, отправляют биты слова по линиям передачи, одновременно с этим получают отправленные другой стороной биты слова по линиям приема. Таким образом происходит обмен словами размером в один байт: и ведущий, и ведомый получают по слову за один цикл передачи.

Интерфейс SPI достаточно распространен, контроллеры SPI часто встраиваются в различные микропроцессоры и микроконтроллеры. Разные устройства, требующие большей скорости передачи чем у I2C, имеют интерфейс SPI. В работе [5] рассматривается возможность управления цифро-аналоговым преобразователем с ПЛИС по интерфейсу SPI.

В работе [6] разработан блок для подключения микропроцессора MicroBlaze к устройствам, имеющим интерфейс SPI. Для подключения разработанного блока к микропроцессору используется шинный интерфейс AXI-Lite[9], разработанный фирмой ARM Holding: микропроцессор MicroBlaze по шине AXI-Lite отправляет запросы блоку, который уже выполняет обмен данными с внешним устройством по шине SPI.

[9]AMBA AXI and ACE Protocol Specification. AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite, URL https://static.docs.arm.com/ihi0022/e/IHI0022E_amba_axi_and_ace_protocol_spec.pdf

Шинный интерфейс AXI-Lite часто используется для объединения различных блоков внутри ПЛИС. Близкий по функциональности блок разработан в работе [7]. В отличие от предыдущей работы он использует шинный интерфейс APB[10], так же разработанный фирмой ARM Holding. Условия работы этих блоков существенно отличаются от требуемых: используются шинные интерфейсы AXI-Lite и APB, а не пакетный интерфейс.

Интерфейс QSPI (Quad SPI) является модификацией интерфейса SPI. Важным отличием является то, что передача ведется не по двум линиям данных, а по четырем. Кроме этого, все четыре линии могут использоваться при передаче в одну сторону. Таким образом достигается учетверенная скорость передачи в одну сторону по отношению к SPI. Интерфейс QSPI используется для подключения микросхем flash-памяти.

Подобный интерфейс используется и для подключения SD-Card flash-памяти. Данные карты памяти могут поддерживать несколько протоколов взаимодействия: стандартный SPI, 1- или 4-битовый SD-режим и использование дифференциальных пар в SD-картах поколения UHS-II. В работе [8] рассматривается разработанный блок для подключения SD-карт к шине APB. Показано, что разработанный блок может функционировать на частоте до 281 МГц, а при частоте 100 МГц обеспечивает теоретическую скорость передачи до 50 МБ/с. При использовании различных карт памяти формата SD реальная скорость передачи составила до 43 МБ/с.

Хотя скорость передачи данных достаточна для нашей задачи, но заложенная в данные протоколы асимметричность (микропроцессор — память) не позволяет их использовать в наших условиях.

Подчеркнем, что важным отличием SPI от UART является то, что используется отдельная линия передачи тактового сигнала (от ведущего к ведомому). Именно это позволяет SPI функционировать на частотах от 1 МГц до 50–66 МГц и даже выше. В результате SPI и QSPI обеспечивают близкую к необходимой скорость передачи. Но они не обеспечивают необходимую нам симметричную пакетную передачу данных.

[10]AMBA Specification (Rev 2.0), URL https://static.docs.arm.com/ihi0011/a/IHI0011A_AMBA_SPEC.pdf

## 2.4. LPC

Шина LPC[11](Low Pin Count, небольшое число контактов) — синхронная шина, разработанная Intel с целью подключения к системе устройств, не требующих большой пропускной способности, таких как загрузочное ПЗУ, последовательный и параллельный порты. Шина создавалась как замена шины ISA для устройств, которым не требуется пропускная способность шины PCI.

Спецификация на интерфейс этой шины определяет шесть обязательных линий, необходимых для организации двусторонней передачи (рисунок 5): тактовый сигнал, сигнал сброса, сигнал начала передачи (фрейма) и четыре линии передачи данных. Существуют еще шесть дополнительных линий (могут отсутствовать в различных реализациях), которые используются для организации DMA (direct memory access, прямой доступ в память), передачи прерываний от ведомых устройств, управления питанием и прочего.

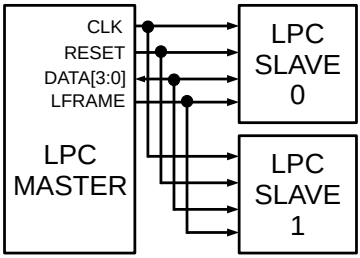


Рисунок 5. Схема интерфейса LPC

Для синхронизации ведущего и ведомых используется тактовый сигнал частотой 33 МГц, генерируемый ведущим. В отличие от UART и SPI, но аналогично QSPI, для передачи данных шина LPC использует четыре двусторонних линии.

Каждая передача (фрейм) по шине LPC — это посылка запроса ведущим устройством и получение ответа от ведомого устройства (рисунок 6). Передача начинается путем взведения специального сигнала начала фрейма. Затем по линиям данных ведущее устройство передает запрос, содержащий тип запроса, адрес и данные (полный формат запросов и ответов описан в спецификации LPC). После этого управление линиями данных передается от ведущего устройства к ведомому, которое отправляет по линиям данных ответ на запрос.

[11]Intel Low Pin Count (LPC) Interface Specification, URL https://www.intel.com/content/dam/www/program/design/us/en/documents/low-pin-count-interface-specification.pdf

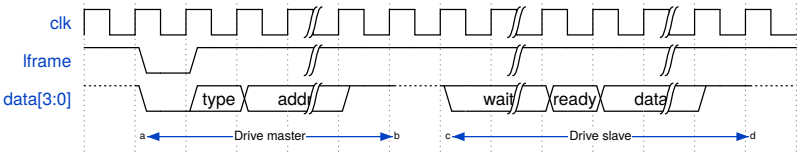


Рисунок 6. Временная диаграмма запроса чтения от ведущего к ведомому в протоколе LPC

Важными для нас особенностями интерфейса LPC являются использование отдельной линии тактового сигнала и поочередное использование двухсторонних линий данных, что обеспечивает необходимую нам скорость передачи. Но жесткая ориентация на несимметричное разделение устройств на ведущее-ведомое не позволяет использовать LPC в разрабатываемой вычислительной системе.

## 2.5. Интерфейс на основе AXI-Lite

Интерфейс AXI-Lite AXI-Lite предназначен для соединения блоков внутри микропроцессоров и ПЛИС. Он имеет 146 линий, что позволяет компактно описывать использующие его блоки. Однако такое количество линий не позволяет использовать AXI-Lite для взаимодействия микросхем между собой.

В работе [9] автор предлагает использовать AXI-Lite для соединения двух ПЛИС, но для уменьшения числа требуемых линий предлагает использовать сериализацию/десериализацию для передачи широкого слова данных по небольшому числу линий. В результате автор использует 19 линий вместо 146: 9 линий для передачи данных в одну сторону, 9 линий для передачи в обратную сторону и одну линию для передачи тактового сигнала (рисунок 7). Использование отдельных линий для передачи от ведущего устройства к ведомому и в обратную сторону позволяют передавать в обе стороны данные независимо, но ценой большого числа линий. Отсутствие сигнала сброса и передача тактового сигнала от ведомого к ведущему являются особенностями вычислительной системы, используемой авторами, не влияющими на применимость данного подхода.

Блоки сериализации/десериализации построены на классическом сдвиговом регистре. Это позволило обойтись без использования специальных блоков памяти. В результате блоки для ведущего и ведомого устройств занимают небольшую часть ПЛИС: 278 LUT[12]

[12]LUT — LookUp Table, таблица поиска: элементарный блок внутри ПЛИС, реализующий логическую функцию (обычно от 4–6 аргументов) на основе таблицы.

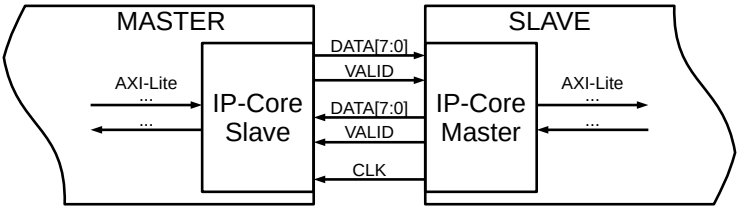


Рисунок 7. Схема интерфейса на основе AXI-Lite

и 511 регистров в ведущем устройства, и 62 LUT и 167 регистров в ведомом устройстве.

Интересно отметить, что в данном интерфейсе между ПЛИС нет ни явных, ни специально закодированных сигналов о готовности принимать данные или о получении данных. Интерфейс AXI-Lite устроен таким образом, что в каждый момент времени может происходить только одна транзакция между ведущим устройством и ведомым. Поэтому до завершения текущей транзакции новые данные не будут переданы. В результате достаточно иметь небольшое число регистров для хранения данных текущей транзакции, которые также можно использовать как сдвиговые регистры.

Использование шины AXI-Lite и 19 линий между ПЛИС не позволяют использовать данную реализацию в разрабатываемой нами вычислительной системе.

## 3. Разработанный интерфейс Tlink

### 3.1. Интерфейс Tlink

На основе проведенного анализа существующих решений, авторами был разработан и реализован интерфейс Tlink с целью оптимального для создаваемой вычислительной системы подключения управляющей ПЛИС к нескольким вычислительным ПЛИС.

Перед описанием собственно интерфейса Tlink между двумя микросхемами ПЛИС, опишем пользовательский интерфейс, по которому другой блок внутри ПЛИС (в дальнейшем такой блок мы будем называть «пользовательским блоком») будет взаимодействовать с блоком Tlink. Пользовательский интерфейс состоит из двух наборов линий (по одному в каждую сторону). Каждый набор состоит из линий передачи данных (‘data’) и стандартных сигналов «рукопожатия»

(‘valid’ и ‘ready’) (рисунок 8). Данный интерфейс является широко используемым, например, он аналогичен интерфейсу AXI-Stream[13], который используется для соединения блоков внутри ПЛИС.

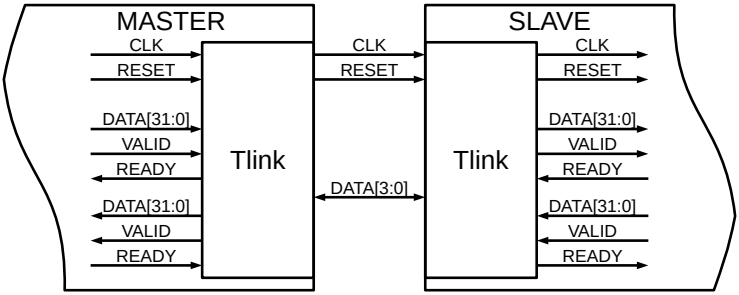


Рисунок 8. Схема интерфейса Tlink

По линиям ‘data’ передаются пакеты, аналогично, например, интерфейсу AXI-Stream. В отличие от других протоколов сигналы начала пакета (‘sop’ — start of packet, начало пакета) и конца пакета (‘eop’ — end of packet, или ‘last’ — последний флит пакета) не передаются, а восстанавливаются исходя из знания размера пакета. Сигналы «рукопожатия» также стандартны: данные по линиям ’data’ считаются переданными на данном такте, если обе линии ’valid’ и ’ready’ равны ’1’ на данном такте.

Отметим существенные особенности использования данного пользовательского интерфейса, на которые опирается наша реализация интерфейса Tlink:

- Если пользовательский блок готов принять пакет, то он должен выставить сигнал ‘ready’, не дожидаясь сигнала ‘valid’ о готовности пакета, и должен держать сигнал ‘ready’ до тех пор, пока не будет полностью передан пакет из Tlink в пользовательский блок.
- Если пользовательский блок готов передать пакет, то он должен выставить сигнал ‘valid’, не дожидаясь сигнала ‘ready’ о готовности блока Tlink к приему данных, и должен держать сигнал ‘valid’ до тех пор, пока не будет полностью передан пакет из пользовательского блока в Tlink.

[13]AMBA 4 AXI4-Stream Protocol. Specification, URL https://static.docs.arm.com/ihi0051/a/IHI0051A_amba4_axi4_stream_v1_0_protocol_spec.pdf

Это позволяет избежать буферизации внутри Tlink и не использовать вспомогательные буфера. При необходимости обойти эти ограничения или для передачи данных между блоками, использующими разные частоты, пользовательский блок может использовать стандартные промежуточные FIFO-буфера (first in, first out — очередь).

Интерфейс Tlink состоит из шести линий: тактовый сигнал (аналогично SPI и LPC), сигнал сброса (аналогично LPC) и четыре двухсторонние линии передачи данных (аналогично QSPI и LPC) (рисунок 8).

Тактовый сигнал передается от ведущего устройства к ведомому и используется для синхронизации реализаций интерфейса Tlink в устройствах. Это позволяет безошибочно передавать данные на частотах до 66–100 МГц.

Сигнал сброса передается от ведущего устройства к ведомому и используется для приведения ведомого устройства к начальному состоянию (сброса всех внутренних регистров).

Четыре двухсторонние линии предназначены для полудуплексной передачи пакетов как от ведущего устройства к ведомому, так и в обратную сторону. Предполагается, что при разработке платы или прошивки ПЛИС эти линии подтянуты к ‘1’, как и в других аналогичных интерфейсах. Это обеспечивает фиксированный уровень ‘1’ на линиях в те моменты времени, когда ни одно устройство не выдает сигнал на линии.

Отметим, что отличия ведущего и ведомого устройств в интерфейсе Tlink заключаются в направлении передачи тактового сигнала и сигнала сброса, а так же в направлении первой передачи пакета после сброса. В дальнейшем при передаче пакетов устройства симметричны.

На уровне линий Tlink близок к рассмотренному выше интерфейсу LPC, но не использует отдельную линию начала передачи пакета, что позволяет немного уменьшить количество используемых линий.

Передача данных ведется пакетами длинной от 4 байт до 128 байт. Каждый пакет содержит заголовок (4 байта) и тело пакета (от 0 байт до 124 байт). Для протокола Tlink в заголовке пакета существенными являются следующие поля: длина пакета (размер этого поля составляет 5 бит) и кредитная информация (1 бит), о которой будет сказано ниже.

Передача пакетами ведется строго по очереди (рисунок 9). Сначала ведущее устройство передает пакет, а ведомое принимает пакет. Затем

наоборот: ведомое устройство передает пакет, а ведущее принимает. Далее этот цикл повторяется до бесконечности.

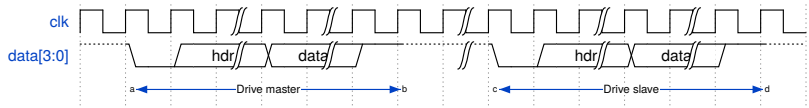


Рисунок 9. Временная диаграмма передачи одного пакета в протоколе Tlink

Начало пакета определяется переходом всех линий данных из уровня ‘1’ в уровень ‘0’ (аналогично UART). Это позволяет не задействовать отдельную линию для обозначения начала пакета. Конец пакета определяется по указанной в заголовке длине пакета.

Отметим, что изменяя параметры реализации, можно настроить интерфейс Tlink исходя из требований различных систем: например, можно задать число линий для передачи данных, ширину поля длины пакета и, соответственно, поддерживаемые длины пакетов, размер заголовка.

Далее рассмотрим реализацию интерфейса Tlink и используемый протокол более подробно.

## 3.2. Структура реализации интерфейса Tlink

Реализация интерфейса Tlink состоит из трех частей:

- ядро интерфейса Tlink, реализующее прием и передачу пакетов, контроль над линиями данных, начальную синхронизацию и выход из некорректных состояний;
- сервисный уровень интерфейса Tlink, который обеспечивает передачу кредитной информации и служебных пакетов;
- «наблюдатель» интерфейса Tlink, который обеспечивает сбор статистики о работе интерфейса.

## 3.3. Ядро интерфейса Tlink

Ядро интерфейса Tlink должно обеспечивать передачу и прием пакетов. Порядок передачи и приема пакетов жестко фиксированы и описываются конечным автоматом, который имеет следующие состояния:

(1) Инициализация.
(2) Ожидание пакета для отправки (WAIT_SERV).
(3) Отправка пакета (SEND_PKT).
(4) 1 этап передачи контроля над линиями данных (TAR1[14]).
(5) 2 этап передачи контроля над линиями данных (TAR2).
(6) Ожидание пакета для приема (WAIT_LINK).
(7) Получение пакета (RECV_PKT).
(8) 1 этап получения контроля над линиями данных (TAR3).
(9) 2 этап получения контроля над линиями данных (TAR4).
(10) Восстановление после ошибки.

Подчеркнем, что на уровне ядра передача асимметрична: есть ведущее и ведомое устройства. Отличия ведущего от ведомого заключаются в следующем: ведущий генерирует тактовый сигнал и сигнал сброса; оба участника передачи переходят в разные состояния из состояния инициализации.

После сброса ведущий переходит из состояния инициализации в состояние WAIT_SERV, а ведомый — в состояние WAIT_LINK. Связано это с тем, что ведущий начинает передачу и первым передает пакет после сброса, в то время как ведомый этот первый пакет получает. Отметим, что этот пакет может не содержать данных и быть служебным.

В состоянии ожидания пакета для отправки (WAIT_SERV) на все четыре линии данных выдается уровень ‘1’. Когда пакет поступил на вход ядра, то на линии данных выдается уровень ‘0’ и ядро переходит в состояние передачи пакета (SEND_PKT). По переходу линий данных из ‘1’ в ‘0’ партнер понимает, что сейчас начнется передача пакета.

В состоянии передачи пакета (SEND_PKT) пакет последовательно передается по линиям данных с использованием сериализатора на сдвиговом регистре (аналогично интерфейсу на основе AXI-Lite). Заголовок пакета содержит информацию о количестве данных в пакете, на основе которой ядро Tlink определяет, когда заканчивать передачу пакета.

После окончания передачи пакета ядро Tlink передает управление партнеру путем последовательного перехода в состояния TAR1 и TAR2. В состоянии TAR1 устройство выводит на линии данных уровень ‘1’, а в состоянии TAR2 перестает выдавать сигналы на линии,

[14]TAR — Turn-ARound, разворачивание линии передачи данных: передача контроля над линией передачи данных от одного устройства к другому.

переводя выходы в высокоимпедансное состояние. Из-за подтяжки линий к ‘1’ уровень на линиях остается постоянным и равным ‘1’. В каждом из состояний TAR1 и TAR2 ядро находится ровно один такт. После состояние TAR2 управление линиями данных переходит к партнеру, ядро переходит в состояние ожидания пакета для получения (WAIT_LINK).

В состоянии ожидания пакета для получения (WAIT_LINK) ядро читает значения на линиях данных. Как только обнаруживается переход из уровня ‘1’ в уровень ‘0’ по всем линиям, то это означает, что далее по линиям данных будет передан пакет, и ядро переходит в состояние получения пакета (RECV_PKT).

В состоянии получения пакета (RECV_PKT) ядро последовательно получает пакет данных с использованием десериализатора на сдвиговом регистре. Заголовок пакета содержит информацию о количестве данных в пакете, на основе которой ядро Tlink определяет, когда заканчивать прием пакета.

После окончания приема пакета ядро Tlink принимает управление путем последовательного перехода в состояния TAR3 и TAR4. В каждом из состояний TAR3 и TAR4 ядро находится ровно один такт. После состояния TAR4 управление линиями данных переходит к данному ядру, ядро переходит в состояние ожидания пакета для отправки (WAIT_SERV).

Использование состояний TAR1, TAR2, TAR3 и TAR4 позволяет плавно передать контроль над линиями данных от одного устройства к другому и исключить ситуацию, когда оба устройства одновременно выдают данные, что может привести к выходу из строя одного из устройств из-за высоких токов на линиях.

### 3.4. Сервисный уровень интерфейса Tlink

Основная задача сервисного уровня интерфейса Tlink — обеспечить симметричную передачу пакетов, а также передачу пакетов только тогда, когда принимающая сторона готова обработать новый пакет.

Для этого требуется передавать по интерфейсу Tlink дополнительную служебную информацию: информацию о готовности каждой стороной принимать данные. Если принимающая сторона не готова принять данные, то необходимо передать ей управление, не передавая данные. Например, если вспомогательные буфера полностью заполнены и не могут вместить новый входящий пакет, то должна быть возможность попросить партнера временно не передавать новые пакеты.

Для передачи управления используются специальные служебные пакеты, состоящие только из заголовка (без данных). В результате все пакеты состоящие только из заголовка обрабатываются сервисным уровнем и не передаются в пользовательский блок. С одной стороны, это запрещает пользователю посылать такие небольшие пакеты, упаковав все необходимые данные в заголовок. С другой стороны, в разрабатываемой авторами системе такие небольшие пакеты не используются.

Для передачи информации о состоянии приемной части интерфейса Tlink используется один «кредитный» бит в заголовке пакета. Он присутствует во всех пакетах, как служебных, так и обычных. Если получатель принимает пакет с установленным «кредитным» битом, то это означает, что он имеет право послать в ответ пакет с данными, и этот пакет будет успешно получен и обработан. Если «кредитный» бит не установлен в пришедшем пакете, то получатель не имеет право послать пакет с данными, а должен послать служебный пакет для передачи управления.

Работа сервисного уровня Tlink жестко фиксирована и реализована на основе конечного автомата. Конечный автомат сервисного уровня имеет четыре состояния:

(*1*) Простой (IDLE).

(*2*) Отправка служебного пакета (SEND_META).

(*3*) Отправка обычного пакета (SEND_PKT).

(*4*) Получение (обычного или служебного) пакета (RECV).

Одной из основных функций сервисной части является организация симметричной передачи данных. Симметричность организуется следующим образом.

После сброса ведущий переходит в состояние отправки служебного пакета (SEND_META), а ведомый в состояние получения пакета (RECV).

После отправки (обычного или служебного) пакета сервисный уровень переходит в состояние получения пакета (RECV).

Как только получен входящий пакет (состояние RECV), то сервисный уровень переходит в состояние посылки пакета. Если другая сторона Tlink готова получать данные (получен пакет с установленным «кредитным» битом) и есть пакет с данными на отправку, то сервисный уровень переходит в состояние посылки обычного пакета (SEND_PKT). Иначе сервисный уровень переходит в состояние посылки служебного

пакета (SEND_META). В обоих случаях в отправляемых пакетах корректно устанавливается «кредитный» бит.

Таким образом, передача пакетов получается симметричной:

- Обе стороны Tlink посылают пакеты поочередно и сразу, как только наступает их очередь.
- Если у пользовательского блока есть пакет с данными и партнер готов его принять, то как ведущий, так и ведомый, отправляют этот пакет. В противном случае передается служебный пакет.
- Из-за того, что в обратную сторону пакет (обычный или служебный) будет отправлен сразу после получения входящего пакета, то отправка пакета не будет заблокирована.

С точки зрения пользовательского блока, использующего блок Tlink, передача пакетов происходит симметричным и неблокирующим образом.

### 3.5. «Наблюдатель» интерфейса Tlink

Третьей частью реализации интерфейса Tlink является блок «наблюдатель». «Наблюдатель» собирает статистику о количестве переданных и принятых служебных пакетах и пакетах с данными. Статистика собирается за указанное в параметрах блока число тактов и затем выдается пользователю.

Сбор статистики позволяет судить на сколько загружен интерфейс Tlink, является ли он узким местом в вычислительной системе.

### 3.6. Сравнение с существующими решениями

Сравнивая Tlink с UART можно выделить основное преимущество UART: он может работать на расстоянии до 15 метров. В первую очередь это достигается за счет низкой частоты. В контексте поставленной задачи такая надежность передачи не требовалась, достаточно было организовать передачу в пределах одной платы. Недостатком UART является отсутствие кредитных механизмов и пакетной передачи, их придется реализовывать поверх этого протокола. Помимо этого, UART имеет всего одну линию передачи в каждом из направлений, в то время как Tlink имеет четыре линии передачи данных. Также в UART не используется отдельная линия для тактового сигнала: на приемной стороне тактовый сигнал нужно подстраивать под принимаемые данные, что составляет определенные трудности при высокой скорости передачи.

По сравнению с SPI интерфейс Tlink имеет меньше «лишних» линий, поскольку в SPI задействована дополнительная линия выбора устройства, которая активно участвует в передаче. В Tlink такой линии нет, поэтому он использует линии более экономно. Кроме того SPI, как и UART, передает данные сразу в двух направлениях, что снижает его пропускную способность при передаче большого потока данных в одном направлении. Отметим, что в QSPI эта проблема отсутствует — в односторонней передаче используются все четыре линии данных. Стоит отметить, что SPI и QSPI асимметричные интерфейсы — передачу в них инициирует ведущий. Другими недостатками этих двух интерфейсов являются небольшой и фиксированный размер пакета и отсутствие поддержки кредитной информации.

Сравнивая интерфейс LPC с Tlink стоит заметить, что LPC использует несколько больше линий. Так для сигнала начала передачи используется отдельная линия, в то время как в Tlink начало передачи обозначается уровнем сигнала на линиях данных. Кроме того, LPC использует дополнительные линии, позволяющие ведомому сообщить о желании передать пакет ведущему. В Tlink такая информация не требуется: ведущий регулярно передает контроль над интерфейсом ведомому, позволяя тому передавать данные. Как и в других интерфейсах, в LPC отсутствует кредитный механизм.

Таким образом, интерфейс Tlink наиболее соответствует таким требованиям к интерфейсу, как минимум линий, использование пакетной передачи и кредитного механизма, симметричность.

Сравнивая интерфейс Tlink с другими полудуплексными протоколами следует отметить, что пакеты в полудуплексном Tlink передаются по очереди: сначала ровно один пакет передается в одну сторону, потом ровно один пакет передается в обратную и так далее. В результате при полной нагрузке на Tlink скорости передачи пакетов в каждом направлении, измеряемые в пакетах в секунду, будут одинаковы. А общая пропускная способность интерфейса Tlink, измеряемая в байтах в секунду, будет делиться между направлениями не поровну, а пропорционально длине пакетов.

В разрабатываемой авторами системе эта особенность не является недостатком: с одной стороны Tlink не будет полностью загруженным, а с другой стороны, ожидается, что количество передаваемых (в единицу времени) пакетов в каждом направлении будут примерно одинаковы.

### 3.7. Измерение пропускной способности

При проведении измерения пропускной способности будем использовать частоту 20 МГц. Предельная теоретическая пропускная способность четырех линий связи составляет 9,54 МБ/с:[15]

$$\text{Freq} \cdot \text{Width} = 20\ \text{МГц} \cdot 4\ \text{бит} = 80000000\ \text{бит/с} = \ 9{,}54\ \text{МБ/с}.$$

Теоретически рассчитаем пропускную способность разработанного интерфейса Tlink в следующих условиях: в одну сторону передаются пакеты, содержащие $N$ байт (не считая заголовок), в обратную сторону данные не передаются, передаются только служебные пакеты.

Тогда число тактов, необходимых для отправки пакета с данными и получения ответа, будет следующим:

$$(2 \cdot (4 + N)) + 2 + 8 + 2 + 5 = 2 \cdot N + 25,$$

где:

- $(2 \cdot (4 + N))$ — такты, необходимые для передачи 4 байт заголовка и $N$ байт тела пакета по четырем линиям связи,
- 2 — такты, необходимые на TAR1 и TAR2,
- 8 — такты, необходимые для передачи служебного пакета,
- 2 — такты, необходимые на TAR3 и TAR4,
- 5 — такты, необходимые для прохождения сигнала через приемные и передающие регистры в ПЛИС (4 такта), дополнительный такт возникает из-за разницы фаз тактового сигнала у передающего и принимающего устройств.

В результате расчетная пропускная способность будет вычисляться по формуле:

$$N \cdot \frac{200000000}{2 \cdot N + 25}\ \text{Б/с} = N \cdot \frac{19,07}{2 \cdot N + 25}\ \text{МБ/с}, \tag{1}$$

где первый множитель $N$ — полезная длина пакета (в байтах), второй множитель — количество отправленных пакетов за одну секунду.

Для практического измерения пропускной способности будем использовать блок «наблюдатель» интерфейса Tlink, а также дополнительный блок в ПЛИС для постоянной генерации и получения пакетов указанной длины. При получении пакетов дополнительно проверяется отсутствие ошибок при передаче.

[15] МБ — Для указания количества информации мы используем двоичные приставки: 1 МБ = 1048576 байт.

Результаты измерения пропускной способности представлены в таблице 1. В первой строке указана полезная длина пакета (не считая заголовка пакета).

Таблица 1. Пропускная способность

| Длина пакета (байт) | 4 | 8 | 16 | 32 | 64 | 124 |
|---|---|---|---|---|---|---|
| Скорость передачи (МБ/с) | 2,31 | 3,72 | 5,35 | 6,86 | 7,98 | 8,66 |
| КПД (%) | 24% | 39% | 56% | 72% | 84% | 91% |

Во второй строке приведена измеренная пропускная способность. Отметим, что так как генерация пакетов и измерения происходили непосредственно в ПЛИС, поэтому никаких накладных расходов не возникает. Поэтому рассчитанная по формуле 1 пропускная способность совпадает с измеренной пропускной способностью.

В третьей строке приведена эффективность (КПД) передачи данных по отношению к теоретической пропускной способности. При пакетах с данными максимальной длины 124 байт КПД составило около 91%, что является хорошим показателем эффективности реализованного подхода.

Дополнительно было проведено исследование надежности разработанного интерфейса Tlink на более высоких частотах 66 и 100 МГц. Оказалось, что для обеспечения безошибочной передачи данных необходимо на ведущем и ведомом устройствах использовать тактовые сигналы одинаковые по частоте, но отличающиеся по фазе. Для этого в ведущее устройство был добавлен блок автоподстройки частоты (PLL, phase-locked loop), который изменял фазу у передаваемого на ведомое устройство тактового сигнала. Конкретная разница фаз зависит от частоты и длины линии связи между устройствами. Планируется продолжить исследования надежной работоспособности интерфейса на высоких частотах.

## Заключение

Разработка вычислительной системы на базе ПЛИС поставила перед авторами задачу разработать протокол взаимодействия микросхем между собой со следующими требованиями:

- поддерживать симметричную неблокируемую пакетную передачу данных,
- использовать до 8 линий ввода-вывода общего назначения,

- обеспечить скорость передачи на уровне 10 МБ/с.

В результате авторами был разработан и реализован пакетный протокол Tlink, обладающий следующими свойствами:

- Протокол обеспечивает двунаправленную передачу пакетов.
- Используются 6 линий ввода-вывода общего назначения (одна линия используется для передачи опорной частоты, одна линия используется для передачи сигнала сброса, четыре линии используются для передачи данных в полудуплексном режиме). Возможно использование другого числа линий для передачи данных путем изменения параметров в реализации.
- Поддерживается передача пакетов длиной до 128 байт. Возможна поддержка пакетов других длин путем изменения параметров в реализации.
- Протокол передачи «знает» только о 6 битах в пакете: 5 бит задают длину пакета и 1 бит используется для передачи кредитной информации. Возможна поддержка пакетов других длин путем изменения параметров в реализации.
- Основная частота передачи 20 МГц, но поддерживаются частоты до 100 МГц (работа на высоких частотах определяется в первую очередь качеством разводки платы и поддержкой высокой частоты в линиях ввода-вывода ПЛИС).
- Получена реальная скорость передачи до 8,66 МБ/с (КПД передачи до 91%). При использовании более высокой частоты передачи скорость будет пропорционально выше. При использовании реализации протокола с более длинными пакетами КПД передачи будет выше.

Реализованный протокол удовлетворяет всем требованиям, выдвинутым разрабатываемой авторами вычислительной системой на базе ПЛИС. Отличается от рассмотренных широко используемых решений поддержкой пакетной передачи данных. Протокол реализован на языке VHDL и используется в разработанной вычислительной системе. Возможен перенос разработанного протокола на другие системы на базе ПЛИС или в специализированные микросхемы.

## Список литературы

[1] R. Tsugami, Y. Shibata. “FPGA implementation of lightweight communication protocol processing for IoT”, *Proceedings of the 12th International Conference on Complex, Intelligent, and Software Intensive Systems*, CISIS-2018, Advances in Intelligent Systems and Computing, vol. **772**, eds. L. Barolli,

N. Javaid, M. Ikeda, M. Takizawa, Springer, Cham, 2019, pp. 506–517. ↑32

[2] R. S. Correa, J. P. David. “Ultra-low latency communication channels for FPGA-based HPC cluster”, *Integration*, **63** (2018), pp. 41–55. ↑32

[3] R. S. S. Kumari, C. Gayathri. “Interfacing of MEMS motion sensor with FPGA using I2C protocol”, 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) (17–18 March 2017, Coimbatore, India), pp. 1–5. ↑36

[4] G. Yoon, J. Kim, G.-Y. Kim, B. Son, H. Yoo. “Multiple RS-485 interface management FPGA design for Power micro-metering”, 2019 10th International Conference on Power Electronics and ECCE Asia (ICPE 2019 — ECCE Asia) (27–30 May 2019, Busan, Korea (South)), pp. 2635–2640. ↑37

[5] Н. Н. Беспалов, Ю. В. Горячкин, Д. В. Тундыков. «Программная реализация управляющего устройства на ПЛИС для обмена данными по интерфейсу SPI», *Научно-технический вестник Поволжья*, 2016, с. 75–78. ↑38

[6] В. С. Старшинов, С. А. Ткачев. «Разработка IP-Core для соединения интерфейсов AXI и SPI с использованием микропроцессорных систем в связке с ПЛИС», *Наука. Технологии. Инновации*, Сборник научных трудов, в 10 т. Т. 1, ред. Д. Н. Достовалов, Новосибирский государственный технический университет, Новосибирск, 2018, с. 110–117. ↑38

[7] R. Nawrath, R. Czerwinski. “FPGA-based implementation of APB/SPI bridge”, International Conference of Computational Methods in Sciences and Engineering 2018 (ICCMSE 2018), AIP Conference Proceedings, vol. **2040**, AIP Publishing, 2018, 4 pp. ↑39

[8] Д. И. Воронков, А. А. Вейков, И. Ю. Сысоев. «Сложнофункциональный блок контроллера карты памяти интерфейса Secure Digital», *Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС)*, 2016, №3, с. 186–190. ↑39

[9] М. И. Чемоданов. «Организация интерфейса для реализации протокола AXI-Lite между ПЛИС фирмы Xilinx», *Информационные технологии в моделировании и управлении: подходы, методы, решения*, Материалы I Всероссийской научной конференции, в 2 т. Т. 2, Качалин Александр Васильевич, 2017, с. 261–268. ↑41

*Поступила в редакцию* 25.11.2019
*Переработана* 18.03.2020
*Опубликована* 25.03.2020

Рекомендовал к публикации *к.ф.-м.н. С. А. Романенко*

*Пример ссылки на эту публикацию:*

И. А. Адамович, Ю. А. Климов. «Пакетный протокол взаимодействия программируемых логических интегральных схем». *Программные системы: теория и приложения*, 2020, **11**:1(44), с. 31–55.
10.25209/2079-3316-2020-11-1-31-55
http://psta.psiras.ru/read/psta2020_1_31-55.pdf

Эта же статья по-английски: 10.25209/2079-3316-2020-11-1-57-78

*Об авторах:*

**Игорь Алексеевич Адамович**

Младший научный сотрудник ИПС им. А. К. Айламазяна РАН. Принимал активное участие в разработке коммуникационных сетей «Паутина» и «3D-тор» суперкомпьютера «СКИФ-Аврора».

0000-0001-9728-3024
e-mail: i.a.adamovich@gmail.com

**Юрий Андреевич Климов**

Старший научный сотрудник ИПМ им. М. В. Келдыша РАН, к.ф.-м.н. Разработчик метода специализации на основе частичных вычислений, принимал активное участие в разработке коммуникационных сетей «Паутина» и «3D-тор» суперкомпьютера «СКИФ-Аврора» и коммуникационного программного обеспечения для сетей SCI и «МВС-Экспресс» суперкомпьютера К-100.

0000-0001-5081-1547
e-mail: yuklimov@keldysh.ru

*Sample citation of this publication:*

Igor A. Adamovich, Yuri A. Klimov. “An FPGA packet communication protocol”. *Program Systems: Theory and Applications*, 2020, **11**:1(44), pp. 31–55. (*In Russian*).
10.25209/2079-3316-2020-11-1-31-55
http://psta.psiras.ru/read/psta2020_1_31-55.pdf

The same article in English: 10.25209/2079-3316-2020-11-1-57-78