УДК 004.713 DOI 10.18522/2311-3103-2023-3-15-25

А.В. Мангушев, В.А. Зыбин, И.Д. Полухин

РАЗРАБОТКА СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ НА ПЛИС ДЛЯ ОБРАБОТКИ ETHERNET-ПАКЕТОВ

Разработана схема буферизации Ethernet-пакетов для аппаратной реализации их обработки на базе ПЛИС. Схема спроектирована на уровне RTL на языке System Verilog в среде разработки Quartus II 13.1. Верификация и моделирование было проведено в среде ModelSim Altera. В качестве целевой платформы была выбрана ПЛИС семейства CycloneIV, располагающаяся на отладочной плате DE2-115. Особое внимание уделено модулям приема и передачи данных, а также реализации аппаратной очереди (FIFO) с возможностью изменения ее содержимого модулем обработки. Схема является параметризованной, позволяет изменять глубину очереди за счет одного параметра без внесения изменений в другие части схемы. Особенностью схемы является возможность добавления любого аппаратного модуля, осуществляющего мониторинг, обработку или шифрование сетевого трафика. Для передачи и приема пакетов применен интерфейс MII, что позволяет использовать любые доступные микросхемы физического уровня для приема и передачи пакетов. Устройство допускает без особых сложностей изменить входной и выходной интерфейс, что увеличивает ее универсальность. В системе не используются проприетарные IP ядра, что делает ее максимально переносимой на ПЛИС различных производителей. К главной особенности схемы можно отнести низкую задержку между приемом и отправкой пакета, определяемой лишь параметрами модуля обработки. Результаты работы можно применить в ходе проектирования устройств, осуществляющих передачу данных с предварительной обработкой. Например, сетевое оборудование (коммутаторы, маршрутизаторы), системы мониторинга и сбора данных.

Плис; fpga; verilog; rtl; сетевой трафик; система массового обслуживания; fifo; mii.

A.V. Mangushev, V.A. Zybin, I.D. Polukhin

DEVELOPMENT OF A QUEUING SYSTEM ON FPGA FOR PROCESSING ETHERNET PACKETS

A scheme for buffering Ethernet packets for hardware implementation of their processing based on FPGA has been developed. The scheme is designed at the RTL level in the System Verilog language in the Quartus II 13.1 development environment. Verification and modeling were carried out in the ModelSim Altera environment. An FPGA of the CycloneIV family, located on the A scheme for buffering Ethernet packets for hardware implementation of their processing based on FPGA has been developed. The scheme is designed at the RTL level in the System Verilog language in the Quartus II 13.1 development environment. Verification and modeling were carried out in the ModelSim Altera environment. An FPGA of the CycloneIV family, located on the DE2-115 debugging board, was chosen as the target platform. Particular attention is paid to data reception and transmission modules, as well as the implementation of a hardware queue (FIFO) with the possibility of changing its contents by the processing module. The scheme is parameterized, it allows you to change the queue depth at the expense of one parameter without making changes to other parts of the scheme. A feature of the scheme is the ability to add any hardware module that monitors, processes or encrypts network traffic. The MII interface is used for transmitting and receiving packets, which allows using any available physical layer chips for receiving and transmitting packets. The device allows you to easily change the input and output interface, which increases its versatility. The system does not use proprietary IP cores, which makes it as portable as possible to FPGAs from various manufacturers. The main feature of the scheme is the low delay between receiving and sending a packet, determined only by the parameters of the processing module. The results of the work can be applied during the design of devices that transmit data with preprocessing. For example, network equipment (switches, routers), monitoring and data collection systems.

Fpga; verilog; rtl; network traffic; mass production system; fifo; mii.

Введение. Для сетевых инженеров крайне важной является задача мониторинга, классификации [1], обработки и анализа [2] сетевого трафика. Существует ряд программных [3], аппаратных и программно-аппаратных [4] решений, например Wireshark [5] или Protosphere [6]. В [7] приводится пример использования программно-аппаратного решения. Однако программные средства при прочих равных условиях являются менее производительными. В [8] рассматривается ряд программных анализаторов трафика, в рамках которых происходит только сбор статистики, без изменения самих пакетов. Именно поэтому в данной работе будет рассматриваться аппаратное решение. В качестве платформы проекта используется ПЛИС компании Altera – DE2-115 [9]. Применение именно ПЛИС позволяет легко переконфигурировать схему устройства. А поскольку мы имеем в распоряжении программируемую логику, то малая задержка распространения сигнала и практически безграничная степень параллелизма позволяют добиться высокой скорости обработки данных.

Описание проектируемой системы. В составе схемы требуется наличие как минимум трех модулей, которые будут обеспечивать прием данных с записью в память, обработку полученных пакетов и отправку, соответственно. В дальнейшем модуль приема будет называться write, модуль обработки – handle, а модуль отправки – send. Принятые данные требуется сохранять, и при этом предоставлять к ним доступ всем трем блокам одновременно, что необходимо для слаженной работы системы. Пакеты необходимо отправлять в том же порядке, в котором они пришли, поэтому логично использовать очередь (FIFO) [10, 11].

Для хранения каждого пакета будет использоваться своя отдельная ячейка (модуль cell), устройство которой представлено на рис. 1.

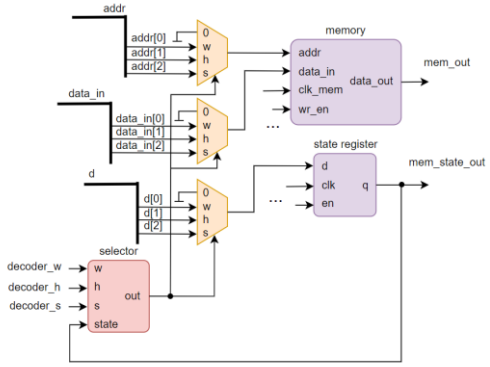

Рис. 1. Устройство модуля cell

Memory – модуль однопортовой памяти [12], в которой и сохраняется пакет. Поскольку максимальный размер пакета в Ethernet ограничен 1500 байтами [13], то объем адресуемой памяти составляет 2048 байт.

State register – регистр состояния ячейки памяти. Ячейка может находиться в одном из трех состояний, закодированных следующим образом: данные отправлены (ячейка готова к записи данных) – 0, данные записаны (ячейка готова к обработке) – 1, данные обработаны (ячейка готова к отправке) – 2. Так как состояний всего 3, то для хранения достаточно двух бит. Тактовый сигнал у регистра и памяти раздельный. Регистр состояния тактируется общим тактовым сигналом, а память от одного из модулей.

К каждому из портов памяти и регистра подключены мультиплексоры, которые предоставляют доступ к ячейке одному из модулей (write, handle или send).

Управление мультиплексорами осуществляет модуль selector. Алгоритм его работы следующий. На входы decoder_w, decoder_h и decoder_s поступают сигналы запроса от соответствующих модулей. При логической единице считается, что соответствующий модуль собирается взаимодействовать с ячейкой. На основании запросов и текущего состояния ячейки происходит следующее: если состояние 0 и запрос на запись, то мультиплексоры предоставляют доступ к памяти модулю write. Иначе если состояние 1 и есть запрос на обработку, то доступ предоставляется модулю handle. Иначе если состояние 2 и запрос на отправку, управление передается модулю send. В противном случае на все входы поступает логический 0, что защищает ячейку и регистр состояния от изменения данных.

Общая схема проекта.

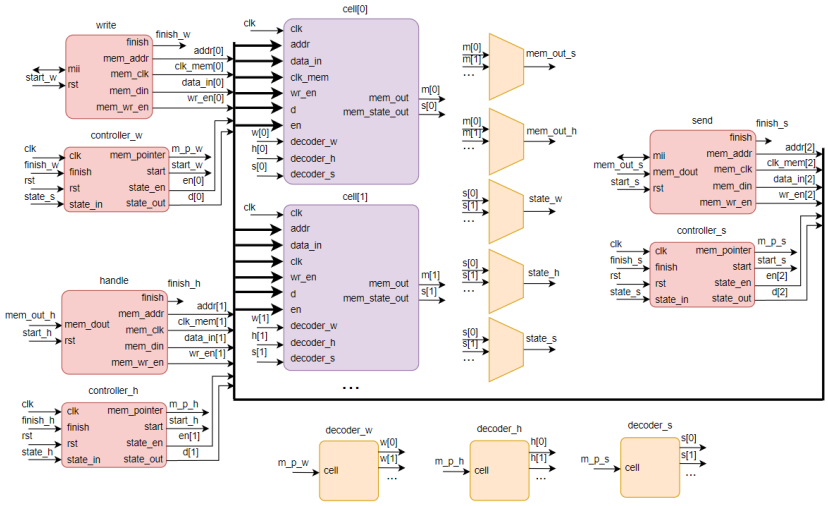

Рис. 2. Общая схема разрабатываемого устройства

На рис. 2 изображена диаграмма, визуализирующая содержимое файла верхнего уровня проекта. Для хранения данных имеется ряд одинаковых ячеек (cell), устройство которых рассмотрено выше. Для обеспечения записи, обработки и отправки данных имеются модули write, handle и send, подробное устройство которых приведено ниже. Каждому из этих модулей требуется свой контроллер, необходимый для выбора ячейки и управления работой модуля. Рассмотрим работу схемы поэтапно.

Изначально после сброса все контроллеры собираются взаимодействовать с ячейкой 0 (cell[0]). Номер ячейки для взаимодействия хранится внутри каждого из контроллеров (controller_w, controller_h, controller_s) и поступает на вход своего декодера (decoder_w, decoder_h, decoder_s). Каждый из них представляет собой преобразователь позиционного кода в унитарный [14]. Логическая единица на соответствующем выходе декодера формирует запрос для соответствующей ячейки. Выбор того, кому предоставить доступ осуществляется внутри ячейки.

Изначально состояние ячейки показывает, что она пуста и готова к записи данных. Поэтому работа с ней предоставляется модулю write. Теперь этот модуль ожидает, когда по интерфейсу MII [15] придет пакет данных. После того как пакет будет принят и записан, будет выставлен сигнал finish_w, после чего controller_w запишет новое состояние в регистр ячейки и начнется отслеживание следующей ячейки. Кроме данных в памяти также сохраняется адрес предпоследнего байта данных, что необходимо для корректного указания граница пакета. Хранение осуществляется в двух первых байтах в порядке от младшего к старшему.

После изменения состояния доступ к данным будет предоставлен модулю handle. Он произведет соответствующую обработку, которая может заключаться в шифровании, изменении полей пакета или внешнем сохранении информации о пакете. В третьем байте памяти кодируется следующая информация: 0 – не отправлять данный пакет, 1 – отправить без пересчета контрольной суммы, 2 – отправить пакет и пересчитать контрольную сумму. После этого выставляется сигнал finish_h и новое состояние записывается в регистр.

Далее управление предоставляется модулю отправки, который первоначально считывает первые три байта и на их основе сразу выставляет сигнал finish_s либо же передает управление соответствующим подмодулям. В конце также производится перезапись состояния ячейки и весь процесс начинается с начала.

Модуль контроллера состояний. Модуль controller (рис. 3) – универсальный параметрический модуль для управления блоком, взаимодействующим с памятью и обеспечивающий взаимодействие с регистром состояний. Описан как конечный автомат Мили, на рис. 4 представлена его диаграмма переходов.

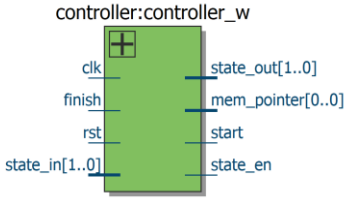

Рис. 3. Блок controller

Вход clk – основной тактового сигнала, rst – общий сброс схемы. На вход state_in поступает текущее состояние отслеживаемой ячейки, на finish – сигнал об окончании работы управляемого модуля. На выходе state_out постоянно находится новое состояние, которое мы будем записывать в регистр состояния, start – управляет подчиненным модулем, если разрешаем его работу, то выставляем логический 1. State_en – вход разрешения записи в регистр состояний. Mem_pointer – адрес текущей отслеживаемой ячейки.

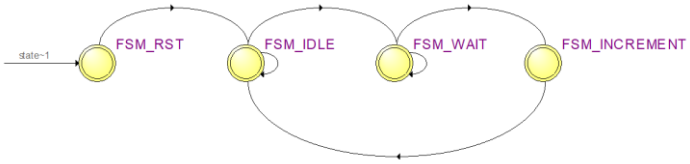

Рис. 4. Состояния конечного автомат модуля controller

После сброса модуль находится в состоянии FSM_RST. В этом состоянии мы сбрасываем адрес отслеживаемой ячейки и переходим в состояние FSM_IDLE. В этом состоянии мы сбрасываем start и state_en в 0 и ожидаем на входе state_in подходящего состояния. Как только это происходит, осуществляем переход в состояние FSM_WAIT, где выставляем на выходе start логическую единицу, активируя подчиненный модуль, и ожидаем логическую 1 на входе finish. При наступлении этого события мы выставляем на start логический 0 и на state_en логическую 1, чтобы записать новое состояние в регистр, после чего попадаем в состояние FSM_INCREMENT. Здесь происходит увеличение указателя на ячейку памяти на 1 и переход в состояние FSM_IDLE, после чего весь цикл повторяется.

Интерфейс MII. Рассмотрим, каким образом происходит прием и передача данных по интерфейсу MII.

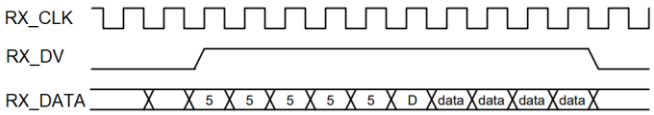

Рис. 5. Сигналы интерфейса MII при приеме данных

Для приема пакета требуется три шины: RX_CLK – тактовый сигнал, RX_DV – сигнал, показывающий валидность данных, RX_DATA – четырехбитная шина данных. Все эти сигналы формируются трансивером.

При начале приема пакета (рис. 5) трансивер выставляет логическую 1 на RX_DV по фронту RX_CLK и на каждый следующий фронт на шину RX_DATA попадает 4 бита данных. Данные поступают в порядке MSB first. Для уведомления об окончании приема, на RX_DV выставляется логический 0 по срезу RX_CLK, после чего пакет считается принятым.

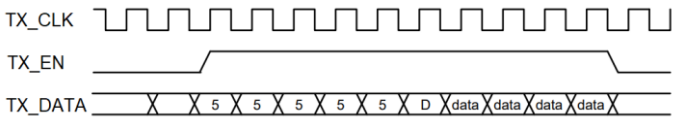

Рис. 6. Сигналы интерфейса MII при отправке данных

Для отправки пакета требуется три шины: TX_CLK – тактовый сигнал, TX_EN – сигнал, показывающий валидность данных, TX_DATA – четырехбитная шина данных. Сигнал TX_CLK поступает от трансивера, NT_EN и TX_DATA управляется устройством, осуществляющим отправку.

Для начала передачи (рис. 6) по фронту TX_CLK выставляем логическую 1 на TX_EN и на каждый следующий фронт на шину TX_DATA отправляем 4 бита данных в порядке MSB first. Для уведомления об окончании передачи, на TX_EN выставляется логический 0 по срезу TX_CLK, после чего пакет считается отправленным.

Модуль записи данных. Модуль write (рис. 7) – осуществляет прием данных по интерфейсу MII и запись в память. Описан как конечный автомат Мура, на рис. 8 представлена его диаграмма переходов.

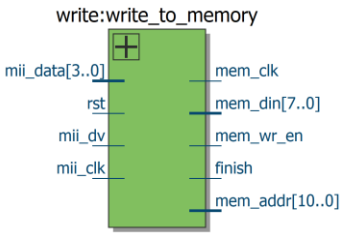

Рис. 7. Модуль write

Вход mii_data – данные от Ethernet трансивера, rst – сброс от блока controller, mii_dv – валидность данных MII, mii_clk – тактовый сигнал от трансивера.

Выход mem_clk – тактовый сигнал для памяти, mem_din – данные для памяти, mem_wr_en – сигнал записи в память, mem_addr – адрес для записи в память, finish – сигнал о завершении работы модуля.

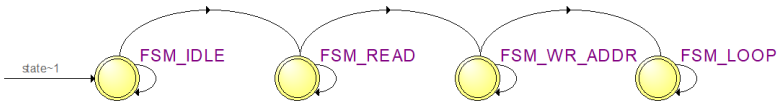

Рис. 8. Состояния конечного автомат модуля write

После сброса контроллер находится в состоянии FSM_IDLE. В этом состоянии ожидает наличия логической 1 на входе mii_dv, после чего переходит в состояние FSM_READ. В этом состоянии производится прием данных и запись в память. Сигналом для перехода в состояние FSM_WR_ADDR является наличие на mii_dv логического 0. Теперь в память производится запись предпоследнего байта с данными и переход к состоянию FSM_LOOP, в котором автомат выставляет в 1 сигнал finish и находится в этом состоянии до прихода сигнала сброса. Такое решение позволяет легче организовать взаимодействие между модулями.

Модуль обработки данных. На данный момент модуль обработки данных не завершен и в качестве демонстрации записывает 0 в определенный байт пакета и указывает что требуется пересчет FCS, поэтому подробное описание не приведено.

Модуль отправки данных. Модуль send (рис. 9) – осуществляет передачу данных из памяти по интерфейсу MII. Описан как конечный автомат Мура, на рис. 10 представлена его диаграмма переходов. Внутри подключается еще два модуля: FCS – для расчета контрольной суммы пакета [16] и mii_tx – для отправки данных по MII.

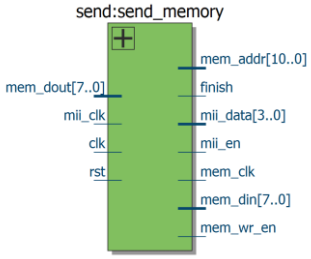

Рис. 9. Модуль send

Вход mem_dout – данные из памяти, mii_clk – тактовый сигнал от Ethernet трансивера, clk – глобальный тактовый сигнал, rst – сброс от блока controller.

Выход mem_addr – адрес для считывания данных, finish – сигнал окончания отправки пакета, mii_data – данные для отправки по MII, mii_en – валидность данных на шине mii_data, mem_clk – тактовый сигнал для памяти, mem_din – данные для записи в память, mem_wr_en – сигнал записи данных в память.

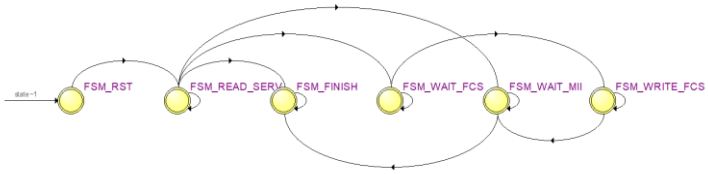

Рис. 10. Состояния конечного автомат модуля send

После сброса контроллер находится в состоянии FSM_RST, где происходит начальная инициализация регистров и переход в состояние FSM_READ_SERV. В этом состоянии производится чтение из памяти предпоследнего адреса данных и

того, что требуется сделать с пакетом. Если пакет не требуется отправлять, то происходит переход в состояние FSM_FINISH, и выставление сигнала окончания обработки. Если же пакет необходимо отправить, то происходит переход в состояние FSM_WAIT_MII, где происходит запуск модуля отправки данных по MII (mii_tx) и по окончании передачи переход в FSM_FINISH. Если требуется пересчитать контрольную сумму, то происходит переход в состояние FSM_WAIT_FSC, в котором запускается модуль для расчета контрольной суммы пакета и после завершения его работы выполняется переход в состояние FSM_WRITE_FCS, где производится перезапись контрольной суммы пакета в память и переход в FSM_WAIT_MII, а далее аналогично описанному.

Модуль интерфейса передачи MII.

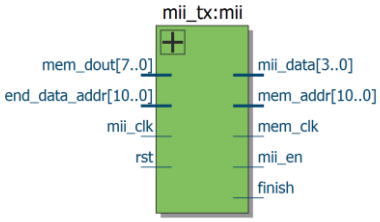

Рис. 11. Модуль mii_tx

Данный модуль производит побайтовое считывание данных из памяти и отправку по интерфейсу MII. Алгоритм работы модуля соответствует описанию из раздела: «интерфейс MII», приведенного выше.

Симуляция схемы. Схема была протестирована с помощью симулятора ModelSim [17]. Для теста было решено использовать 2 ячейки. Тестовая схема аналогична рис. 2. Рассмотрим подробнее результаты симуляции.

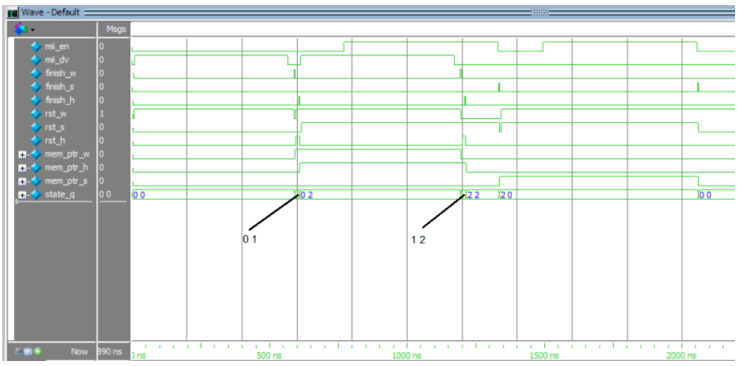

Рис. 12. Результат симуляции в ModelSim

Сигналы finish_w, finish_h и finish_s сигнализируют о выполнении работы модулем записи, обработки и отправки соответственно. Сигналы rst_w, rst_h и rst_s являются сигналами сброса для соответствующих модулей. Когда эти сигналы в состоянии логического 0 – модули неактивны. Сигналы mem_ptr_w, mem_ptr_h и mem_ptr_s – указатели на отслеживаемые ячейки данных для соответствующих модулей. state_q – неупакованный массив с состояниями ячеек.

Изначально все модули, кроме модуля записи находятся в сбросе. Как только входе mii_dv появляется логическая 1 (~50 нс), то начинается запись данных в нулевую ячейку. По появлению импульса в сигнале finish_w (~600 нс) видно момент окон-

чания записи, после чего в ячейку записывается состояние 1 – данные записаны (регистр state_q). Далее свою работу начинает модуль обработки, однако из-за своей простоты крайне быстро ее заканчивает, и состояние сменяется на 2 – данные обработаны, после чего начинается отправка. Однако перед отправкой требуется пересчитать контрольную сумму, отчего присутствует запаздывание в передаче, начало которой можно идентифицировать по состоянию логической 1 на выходе mii_en (~ 750 нс).

В момент завершения обработки данных (~ 600 нс) приходит второй пакет, который будет записан во вторую ячейку памяти. К моменту окончания записи данные из первой ячейки все еще не были отправлены, поэтому ее состояние 2, а состояние второй ячейки сменилось на 1, после чего началась обработка этого пакета. К моменту окончания обработки (~ 1200 нс), модуль отправки данных все еще был занят отправкой данных с первой ячейки, и только после завершения этой операции приступил к отправке данных со второй ячейки. Если подать новый пакет на вход, то он будет записан в первую ячейку, поскольку она свободна и указатель сбросился в 0 после переполнения.

Выводы. В результате был получен код на языке System Verilog, позволяющий реализовать аппаратную платформу для обработки пакетов. Была протестирована возможность захвата пакета с последующей отправкой. Так как в ходе разработки не были применены проприетарные IP ядра [18, 19], есть возможность запустить данный проект практически на любой ПЛИС, имеющей встроенную память. В качестве интерфейса для приема и передачи пакетов был применен MII. Полученные наработки можно применить для доработки схемы под задачи обработки, мониторинга или шифрования [20].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Болдырихин Н.В., Алтунин Ф.А., Короченцев Д.А*. Особенности классификации зашифрованного сетевого трафика // Известия ЮФУ. Технические науки. – 2020. – № 3 (213). – С. 89-98. – DOI: 10.18522/2311-3103-2020-3-89-98.
2. *Белоусов А.С., Будылдина Н.В.* Анализ сетевого трафика: от анализа пакетов до анализа потоков // Инфокоммуникационные технологии: актуальные вопросы цифровой экономики: Сб. научных трудов II Международной научно-практической конференции, Екатеринбург, 26–27 января 2022 г. / под ред. В.П. Шувалова. Сост. М.П. Карачарова. – Екатеринбург: Уральский технический институт связи и информатики (филиал) федерального государственного образовательного бюджетного учреждения высшего профессионального образования "Сибирский государственный университет телекоммуникаций и информатики", 2022. – С. 17-21.
3. *Яковлев Д.А., Синева И.С.* Построение виртуализированной системы фильтрации поддельных сетевых пакетов с использованием Intel DPDK // T-Comm: Телекоммуникации и транспорт. – 2016. – Т. 10, № 8. – С. 30-35.
4. *Лапшичев В.В*. Реализация анализа трафика сети Tor на базе Mikrotik и Suricata // Фундаментальные и прикладные аспекты компьютерных технологий и информационной безопасности: Сб. статей VII Всероссийской научно-технической конференции, Таганрог, 05–11 апреля 2021 г. – Таганрог: ЮФУ, 2021. – С. 58-60.
5. Анализатор трафика для компьютерных сетей. – URL: https://www.wireshark.org/ (дата обращения: 05.05.2023).
6. Protosphere: система анализа сетевого трафика. – URL: https://www.ispras.ru/technologies/protosphere/.
7. *Константинов И.В., Фирсова А.А., Николаева А.В*. Инструмент анализа сетевого трафика // Аллея науки. – 2022. – Т. 1, № 5 (68). – С. 791-794.
8. *Ларин Д.В., Гетьман А.И.* Средства захвата и обработки высокоскоростного сетевого трафика // Тр. Института системного программирования РАН. – 2021. – Т. 33, № 4. – С. 49-68. – DOI: 10.15514/ISPRAS-2021-33(4)-4.
9. DE2 – 115 User manual. – URL: http://www.terasic.com.tw/attachment/archive/502/DE2_115_User_manual.pdf.

10. *Род Стивенс*. Алгоритмы. Теория и практическое применение. – М.: Изд-во «Э», 2016. – 544 с.
11. *Кузьмичев А.М., Рахимьянов А.С.* Формирование и передача пакетов информации по высокоскоростному каналу связи // Механика, управление и информатика. – 2009. – № 1. – С. 495-502.
12. *Соловьев В.* Логическое проектирование встраиваемых систем на FPGA. Ч. 14. Проектирование встроенной памяти в системе Quartus // Компоненты и технологии. – 2019. – № 11 (220). – С. 38-46.
13. IEEE Standard for Ethernet. – URL: https://standards.ieee.org/ieee/802.3/7071/ (дата обращения: 05.05.2023).
14. Цифровой синтез: практический курс / под общ. ред. А.Ю. Романова, Ю.В. Панчула. – М.: ДМК Пресс, 2020. – 556 с.
15. Media-independent interface. – URL: https://en.wikipedia.org/wiki/Media-independent_interface (дата обращения: 05.05.2023).
16. Проект «Марсоход». – URL: https://marsohod.org/projects/marsohod2/263-rtl-recv (дата обращения: 05.05.2023).
17. *Garcна Valderas M., Zumel P., Lбzaro A. [et al.]*. ModelSim-PSIM mixed signal simulation for power electronics digital control design // VLSI Circuits and Systems IV, Dresden, Germany, 04 May 2009. Vol. 7363. – Dresden, Germany, 2009. – P. 73630V-7. – DOI: 10.1117/12.822051.
18. *Бруно Ф.* Программирование FPGA для начинающих / пер. с англ. С.Л. Плехановой / под науч. ред. А.Ю. Романова, Ю.В. Ревича. – М.: ДМК Пресс, 2022. – 304 с.
19. *Морозов И.А.* Интеграция IP-ядер для ПЛИС в реконфигурируемых вычислительных системах // Суперкомпьютерные технологии (СКТ-2016): Матер. 4-й Всероссийской научно-технической конференции, Дивноморское, 19–24 сентября 2016 г. Т. 1. – Дивноморское: ЮФУ, 2016. – С. 74-77.
20. *Мангушев А.В*. Модуль передачи данных по сети Ethernet на базе ПЛИС // XXVII Региональная конференция молодых ученых и исследователей Волгоградской области: Сб. материалов конференции, Волгоград, 02–15 ноября 2022 г. / Редколлегия: С.В. Кузьмин (отв. ред.) [и др.]. – Волгоград: Волгоградский государственный технический университет, 2022. – С. 239-240.

REFERENCES

1. *Boldyrikhin N.V., Altunin F.A., Korochentsev D.A.* Osobennosti klassifikatsii zashif-rovannogo setevogo trafika [Features of the classification of encrypted network traffic], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2020, No. 3 (213), pp. 89-98. DOI: 10.18522/2311-3103-2020-3-89-98.
2. *Belousov A.S., Budyldina N.V.* Analiz setevogo trafika: ot analiza paketov do analiza potokov [Network traffic analysis: from packet analysis to flow analysis], *Infokommunikatsionnye tekhnologii: aktual'nye voprosy tsifrovoy ekonomiki: Sb. nauchnykh trudov II Mezhdunarodnoy nauchno-prakticheskoy konferentsii, Ekaterinburg, 26–27 yanvarya 2022 g.* [Infocommunication technologies: current issues of the digital economy: Collection of scientific papers of the II International Scientific and Practical Conference, Ekaterinburg, January 26–27, 2022], ed. by V.P. Shuvalova. Compiled by M.P. Karacharova. Ekaterinburg: Ural'skiy tekhnicheskiy institut svyazi i informatiki (filial) federal'nogo gosudarstvennogo obrazovatel'nogo byudzhetnogo uchrezhdeniya vysshego professional'nogo obrazovaniya "Sibirskiy gosudarstvennyy universitet telekommunikatsiy i informatiki", 2022, pp. 17-21.
3. *Yakovlev D.A., Sineva I.S.* Postroenie virtualizirovannoy sistemy fil'tratsii poddel'nykh setevykh paketov s ispol'zovaniem Intel DPDK [Building a virtualized system for filtering fake network packets using Intel DPDK], *T-Comm: Telekommunikatsii i transport,* 2016, Vol. 10, No. 8, pp. 30-35.
4. *Lapshichev V.V.* Realizatsiya analiza trafika seti Tor na baze Mikrotik i Suricata [Implementation of traffic analysis of the Tor network based on Mikrotik and Suricata], *Fundamental'nye i prikladnye aspekty komp'yuternykh tekhnologiy i informatsionnoy bezopasnosti: Sb. statey VII Vserossiyskoy nauchno-tekhnicheskoy konferentsii, Taganrog, 05–11 aprelya 2021 g*. [Fundamental and applied aspects of computer technology and information security: Collection of articles of the VII All-Russian Scientific and Technical Conference, Taganrog, April 05–11, 2021]. Taganrog: YuFU, 2021, pp. 58-60.

5. Analizator trafika dlya komp'yuternykh setey [Traffic analyzer for computer networks]. Available at: https://www.wireshark.org/ (accessed 05 May 2023).
6. Protosphere: sistema analiza setevogo trafika [Protosphere: network traffic analysis system]. Available at: https://www.ispras.ru/technologies/protosphere/.
7. *Konstantinov I.V., Firsova A.A., Nikolaeva A.V.* Instrument analiza setevogo trafika [Network traffic analysis tool], *Alleya nauki* [Alley of Science], 2022, Vol. 1, No. 5 (68), pp. 791-794.
8. *Larin D.V., Get'man A.I.* Sredstva zakhvata i obrabotki vysokoskorostnogo setevogo trafika [Tools for capturing and processing high-speed network traffic], *Tr. Instituta sistemnogo programmirovaniya RAN* [Proceedings of the Institute for System Programming of the Russian Academy of Sciences], 2021, Vol. 33, No. 4, pp. 49-68. DOI: 10.15514/ISPRAS-2021-33(4)-4.
9. DE2 – 115 User manual. Available at: http://www.terasic.com.tw/attachment/ archive/502/DE2_115_User_manual.pdf.
10. *Rod Stivens.* Algoritmy. Teoriya i prakticheskoe primenenie [Algorithms. Theory and practical application]. Moscow: Izd-vo «E», 2016, 544 p.
11. *Kuz'michev A.M., Rakhim'yanov A.S.* Formirovanie i peredacha paketov informatsii po vysokoskorostnomu kanalu svyazi [Formation and transmission of information packets over a high-speed communication channel], *Mekhanika, upravlenie i informatika* [Mechanics, control and informatics], 2009, No. 1, pp. 495-502.
12. *Solov'ev V.* Logicheskoe proektirovanie vstraivaemykh sistem na FPGA. Ch. 14. Proektirovanie vstroennoy pamyati v sisteme Quartus [Logic design of embedded systems on FPGA. Part 14. Designing built-in memory in the Quartus system], *Komponenty i tekhnologii* [Components and technologies], 2019, No. 11 (220), pp. 38-46.
13. IEEE Standard for Ethernet. Available at: https://standards.ieee.org/ieee/802.3/7071/ (accessed 05 May 2023).
14. Tsifrovoy sintez: prakticheskiy kurs [Digital synthesis: a practical course], under the general. ed. A.Yu. Romanova, Yu.V. Panchula. Moscow: DMK Press, 2020, 556 p.
15. Media-independent interface. Available at: https://en.wikipedia.org/wiki/Media-independent_interface (accessed 05 May 2023).
16. Proekt «Marsokhod» [Project "Mars rover"]. Available at: https://marsohod.org/ projects/marsohod2/263-rtl-recv (accessed 05 May 2023).
17. *Garcна Valderas M., Zumel P., Lбzaro A. [et al.].* ModelSim-PSIM mixed signal simulation for power electronics digital control design, *VLSI Circuits and Systems IV, Dresden, Germany, 04 May 2009*. Vol. 7363. Dresden, Germany, 2009, pp. 73630V-7. DOI: 10.1117/12.822051.
18. *Bruno F.* Programmirovanie FPGA dlya nachinayushchikh [FPGA programming for beginners], transl. from engl. S.L. Plekhanovoy, under scientific. ed. A.Yu. Romanova, Yu.V. Revicha. Moscow: DMK Press, 2022, 304 p.
19. *Morozov I.A.* Integratsiya IP-yader dlya PLIS v rekonfiguriruemykh vychislitel'nykh sistemakh [Integration of IP cores for FPGAs in reconfigurable computing systems], *Superkomp'yuternye tekhnologii (SKT-2016): Mater. 4-y Vserossiyskoy nachno-tekhnicheskoy konferentsii, Divnomorskoe, 19–24 sentyabrya 2016 g.* [Supercomputer Technologies (SKT-2016): Proceedings of the 4th All-Russian Scientific and Technical Conference, Divnomorskoye, September 19–24, 2016]. Vol. 1. Divnomorskoe: YuFU, 2016, pp. 74-77.
20. *Mangushev A.V.* Modul' peredachi dannykh po seti Ethernet na baze PLIS [FPGA-based Ethernet data transmission module], *XXVII Regional'naya konferentsiya molodykh uchenykh i issledovateley Volgogradskoy oblasti: Sb. materialov konferentsii, Volgograd, 02–15 noyabrya 2022 g.* [XXVII Regional Conference of Young Scientists and Researchers of the Volgograd Region: Conference Proceedings, Volgograd, November 02–15, 2022], ed. board: S.V. Kuz'min (managing ed.) [and others]. Volgograd: Volgogradskiy gosudarstvennyy tekhnicheskiy universitet, 2022, pp. 239-240.

Статью рекомендовал к опубликованию д.т.н., профессор А.В. Боженюк.

Мангушев Александр Вячеславович – Волгоградский государственный технический университет; e-mail: mangushev2001@yandex.ru; г. Волгоград, Россия; тел.: +79880522090; студент.

Зыбин Валерий Андреевич – e-mail: vazybin@mail.ru; тел.: +79610573464; студент.

Полухин Игорь Дмитриевич – e-mail: poluxin.2001@mail.ru; тел.: +79020951311; студент.

Mangushev Alexander Vyacheslavovich – Volgograd State Technical University; e-mail: mangushev2001@yandex.ru; Volgograd, Russia; phone: +79880522090; undergraduate student.

Zybin Valery Andreevich – e-mail: vazybin@mail.ru; phone: +79610573464; undergraduate student.

Polukhin Igor Dmitrievich – e-mail: poluxin.2001@mail.ru; phone: +79020951311; undergraduate student.

УДК 004.5 DOI 10.18522/2311-3103-2023-3-25-35

Д.Е. Чикрин, К.Р. Смольникова

ОБЗОР КОЛЛАБОРАТИВНЫХ РОБОТЕХНИЧЕСКИХ СИСТЕМ И ЮРИДИКО-СИСТЕМНЫЕ АСПЕКТЫ ВЗАИМОДЕЙСТВИЯ С НИМИ

Значительный интерес для отрасли робототехники является исследование многодисциплинарной области – взаимодействие человека и робота (Human-robot interaction, HRI). Индустрия 4.0 (4IR) диктует интенсивное внедрение роботехнических решений во все отрасли экономики и процессы жизнедеятельности людей. Именно поэтому взаимодействие оператора и кобота является одной из самых актуальных тем, влияющая на экономику, рынок труда и общество в целом. На текущий момент кобототехника является одним из новых прорывных направлений в робототехнике, а в связи с развитием стандартов 4IR коботы имеют ключевое преимущество в рамках автоматизации, где полное замещение человеческого труда невозможна. Такая коллаборация навыков оператора и коллаборативного робота ускорит производственно-технологический процесс и позволит компаниям, интегрирующих коботов, стать более конкурентоспособнее, а также свести к минимуму процесс производственных задач. Целью исследования является описание роботехнических систем и анализ юридико-системных аспектов взаимодействия кобота и оператора в совместном рабочем пространстве (collaborative workspace). Задачами исследования являются: 1) общий обзор коллаборативных роботехнических систем по типам: решаемых задач, выполняемых работ и управления; 2) рассмотрение существующих систем оценки рисков при взаимодействии оператора и кобота. Реализация поставленных задач внесет свой вклад в дальнейшие исследования инновационной области HRI, направленная на создание среды для безопасной и эффективной коллаборации оператора и кобота. Практическая ценность настоящей статьи заключается также в системном подходе к рассмотрению сферы кобототехники для дальнейшего изучения безопасных сценариев взаимодействия. По нашему мнению, наиболее эффективным подходом является анализ каждого конкретного случая использования какого-либо вида роботов. Одновременно отмечаем, что в текущих реалиях быстрорастущего сектора робототехники затруднительно классифицировать и унифицировать коллаборативные роботехнические системы в единый акт.

Коллаборативные роботы; взаимодействие человека и робота; коллаборативные роботехнические системы; Индустрия 4.0.

D.E. Chikrin, K.R. Smolnikova

REVIEW OF COLLABORATIVE ROBOTIC SYSTEMS AND LEGAL-SYSTEM ASPECTS OF INTERACTION WITH THEM

Of significant interest to the robotics industry is the study of the multidisciplinary field of human-robot interaction (HRI). Industry 4.0 (4IR) dictates the intensive implementation of robotic solutions in all sectors of the economy and human life processes. That is why the interaction between operator and cobot is one of the most relevant topics affecting the economy, labor market and society as a whole. Currently, cobotics is one of the new breakthrough areas in robotics, and due to the development of 4IR standards, cobots have a key advantage in automation, where full replacement of human labor is impossible. This collaboration of operator and collaborative robot