ТАЛЛИНСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Факультет информационных технологий

IA40LT

Алексей Обухов 143606

РЕАЛИЗАЦИЯ ИГРЫ «ЗМЕЙКА» НА ZEDBOARD

Бакалаврская работа

Руководитель: Александр Судницын
Доцент

Таллинн 2017

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

IA40LT

Aleksei Obuhov 143606

USSIMÄNGU REALISATSIOON ZEDBOARD BAASIL

Bakalaureusetöö

Juhendaja: Aleksander Sudnitsõn
Dotsent

Tallinn 2017

Авторская декларация

Подтверждаю, что данная работа была подготовлена самостоятельно и ни кем ранее не была предоставлена для защиты. На все используемые при составлении работы данные из литературных источников, публикаций и исследовательских работ других авторов приведена ссылка.

Автор: Алексей Обухов

22.05.2017

BAKALAUREUSETÖÖ ÜLESANNE

Üliõpilane: Aleksei Obuhov

Üliõpilaskood: 143606 IASB

Lõputöö teema eesti keeles: Ussimängu realisatsioon ZedBoard baasil

Lõputöö teema inglise keeles: Realization of Snake game on ZedBoard

Juhendaja (nimi, töökoht, teaduslik kraad, allkiri): Aleksander Sudnitsõn, TTÜ Arvutisüsteemide instituut, Ph.D.

Konsultandid:

Lahendatavad küsimused ning lähtetingimused: Realisatsioon peab plema tehtud Zynq kiibi baasil; VHDL projekteerimiskeel; Vivado arenduskeskkond (Xilinx); Prototüübi (maketi) valmistamine.

Eritingimused: Töö vene keeles

Nõuded vormistamisele: Vastavalt Infotehnoloogia teaduskonnas kehtivatele nõuetele

Lõputöö esitamise tähtaeg: 22.05.2017

Ülesande vastu võtnud: _____ kuupäeva:

Аннотация

Целью данной работы является разработка игровой системы с игрой «Змейка» на отладочной плате ZedBoard. Используемая в работе отладочная плата ZedBoard, базой которой является ПЛИС семейства Zynq-7000, предоставлена Институтом Вычислительной техники при Таллиннском Техническом Университете.

Дополнительной целью работы является разработка игрового контроллера для расширения данной игровой системы. Игровой контроллер выполнен с использованием 8 – битного AVR микроконтроллера Atmega328p-pu фирмы Atmel.

В основе работы лежит совместная разработка (*co-design*) аппаратного и программного обеспеченья.

Результатом данной работы является работающая гибкая игровая система с игрой «Змейка», которая в дальнейшем может быть расширена путем дополнения или изменения элементов системы.

Данная работа написана на русском языке и содержит 40 страниц текста, 4 части, 14 рисунков и 2 таблиц, CD-диск с файлами проекта в Vivado 2015.4 и Atmel Studio 7.0.

Annotatsioon

Ussimängu realisatsioon ZedBoard baasil

Käesoleva töö eesmärgiks on ussimänguga mängusüsteemi loomine ZedBoard arendusplaadi baasil. Töös kasutatud plaat on Tallinna Tehnikaülikooli Arvutisüsteemide instituudi ZedBoard arendusplaat, mis põhineb Xilinx Zynq-7000 süsteemkiibil.

Teiseks eesmärgiks on koos selle mängusüsteemiga töötava mängupuldi arendamine ja realiseerimine.

Töö käigus tutvutakse põhjalikult kasutatava riistvaraga. Selgitatakse mängusüsteemi ülesehitust ja töö põhimõtteid; arendatakse süsteemi osad programmeeritava loogika tasemel IP-ploki kujul; realiseeritakse video kontroller, video mälu ja VGA analoog liides; organiseeritakse andmevahetus programmiloogika ja protsessori vahel AXI siini kaudu; kirjutatakse ussimäng; projekteeritakse mängupult ja luuakse I2C protokollil põhinev kommunikatsiooniliides. IP-plokid on kirjutatud VHDL riistvarakirjelduskeeles. Ussimängu ja mängupuldi püsivara on kirjutatud C programmeerimiskeeles.

Töö tulemuseks on töötav mängusüsteem ussimänguga ja mängupult, mis suudab toimida koos selle mängusüsteemiga. Süsteem on paindlik ja seda saab hõlpsasti laiendada lisades vajadusel uusi alamsüsteeme või muutes olemasolevaid.

Lõputöö on kirjutatud vene keeles ning sisaldab teksti 40 leheküljel, 4 peatükki, 14 joonist, 2 tabelit, CD-ketas projekti Vivado 2015.4 ja Atmel Studio 7.0 failidega.

Abstract

Realization of Snake game on ZedBoard

The purpose of the thesis was to develop the realization of the Snake videogame system on development board ZedBoard. ZedBoard is based on Xilinx Zynq-7000 System-on-Chip, which are available in the Department of Computer Engineering in Tallinn University of Technology.

Another objective of this topic was the development and implementation of the ad hoc gamepad for the established game system. Integration of the gamepad has brought minor changes in the game system.

Through the process of implementation, utilized hardware was thoroughly scrutinized. To achieve the desired goals, the main structure of the game systems were examined, IP-blocks used in the game systems of programming logic were created, the video controller, the video memory and the VGA analog interface was implemented, communication between programming logic and processor system by AXI bus were established, Snake game for game system was written from scratch, gamepad was designed and communication protocol between the game system and gamepad was created based on I2C protocol. IP-blocks were written in VHDL hardware description language. Snake game and firmware of gamepad were written in C programming language.

As the result, the game system with Snake videogame and gamepad for this game system was created. Game system is flexible and can be easily extended by adding new subsystems or modification of current subsystems given the necessity.

This thesis is written in Russian and is 40 pages long, including 4 chapters, 14 figures and 2 tables, CD-disc containing project files for Vivado 2015.4 and Atmel Studio 7.0.

Используемые термины и сокращения

VHDL	*Very high speed integrated circuits Hardware Description Language*, язык описания аппаратуры интегральных схем
ARM	*Advanced RISC Machine*, усовершенствованная RISC-машина
RISC	*Reduced Instruction Set Computer*, компьютер с сокращённым набором команд
ПЛИС	Программируемая Логическая Интегральная Схема
VGA	Video Graphics Array, компонентный видеоинтерфейс
AVR	*modified Harvard architecture machine*, архитектура микроконтроллера
AXI	*Advanced eXtensible Interface*, продвинутый расширяемый интерфейс
I2C	*Inter-Integrated Circuit*, схема внутренней связи
IP-блок	*Intellectual Property block*, блок интеллектуальной собственности
ПО	Программное Обеспечение
SoC	*System-on-a-Chip*, система на кристалле
DDR2 (SDRAM)	*Double-Data-Rate 2 Synchronous Dynamic Random Access Memory*, синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных второго поколения
DDR3 (SDRAM)	*Double-Data-Rate 3 Synchronous Dynamic Random Access Memory*, синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных третьего поколения
LPDDR2	*Low Power DDR*, низкопотребляемых интерфейсов DDR
USB	*Universal Serial Bus*, универсальная последовательная шина
OTG	*On-The-Go*, поддержка технологии лёгкого соединения периферийных USB-устройств друг с другом
SD	*Secure Digital*, формат карт-памяти
SDIO	*Secure Digital Input/Output*, стандарт поддержки периферии слотом формата SD

UART	*Universal Asynchronous Receiver-Transmitter*, Универсальный асинхронный приёмопередатчик
CAN	*Controller Area Network*, стандарт промышленной сети
SPI	*Serial Peripheral Interface*, последовательный периферийный интерфейс
GPIO	*General-Purpose Input/Outpu*t, интерфейс ввода/вывода общего назначения
DSP	*Digital Signal Processing*, цифровая обработка сигналов
JTAG	*Joint Test Action Group*, специализированный аппаратный интерфейс на базе стандарта IEEE 1149.1
IEEE	*Institute of Electrical and Electronics Engineers*, Институт инженеров электротехники и электроники
FMC	*FPGA Mezzanine Card*, стандарт подключения дополнительных модулей FPGA
Pmod	*Peripheral Module interface*, интерфейс модулей переферии
XADC	*Xilinx Analog-to-Digital Converter*, стандарт аналогово-цифрового конвертера фирмы Xilinx
HDMI	*High Definition Multimedia Interface*, интерфейс для мультимедиа высокой чёткости
OLED	*Organic Light-Emitting Diode*, органический светодиод
SRAM	*Static Random Access Memory*, статическая память с произвольным доступом
CMOS	*Complementary Metal-Oxide-Semiconductor*, комплементарная структура металл-оксид-полупроводник
EEPROM	*Electrically Erasable Programmable Read-Only Memory*, электрически стираемое перепрограммируемое постоянное запоминающее устройство
MIPS	*Million Instructions Per Second*, миллион инструкций за секунду (мера быстродействия процессора компьютера)
SDK	*Software Development Kit*, комплект средств разработки
BRAM	*Block Random Access Memory*, блок памяти с произвольным доступом
RAM	*Random Access Memory*, память с произвольным доступом
RGB	*Red-Green-Blue*, красный-зеленый-синий
ICSP	*In-Circuit Serial Programming*, внутрисхемное последовательное программирование

Содержание

Перечень таблиц

Перечень рисунков

Введение

Тема данной работы выбрана автором исходя из интереса реализации многоуровневой системы и большого количества возможностей для реализации поставленной задачи. Также важным решающим фактором являлось то, что данная работа подразумевает разработку как на аппаратном уровне, так и на программном. Следует отметить, что мотивацией в данной работе является универсальность примененных решений, которые могут быть использованы в будущем.

Целью данной работы является реализация игровой системы с игрой «Змейка» (“*Snake”*) на отладочной плате ZedBoard, а также проектирование и реализация игрового контроллера на базе микроконтроллера Atmega328p-pu. Используемая в работе отладочная плата ZedBoard предоставлена Институтом Компьютерной техники при Таллиннском Техническом Университете. Основными задачами данной работы являются: изучение структуры игровых систем; разработка элементов игровой системы; организация взаимодействия разных уровней системы; написание игры «Змейка»; проектирование и последующее изготовление игрового контроллера; разработка встроенного программного обеспеченья (*firmware*) для игрового контроллера; программирование игрового контроллера.

Работа поделена на 4 части. Первая часть посвящена обзору используемого оборудования. В данной части приводятся описание используемых устройств и их основные характеристики.

Во второй части описываются основные положения системы «игровая приставка». Также в данной главе приводится описание структуры разрабатываемой системы.

В третьей части описывается реализация игровой системы, создание основных её элементов и написание игры «Змейка» для данной системы.

В четвертой части описан процесс расширения игровой системы путем разработки, реализации и интеграции игрового контроллера в разработанную игровую систему.

1 Обзор используемого оборудования

В данной работе используется отладочная плата ZedBoard фирмы DIGILENT, базой которой является микросхема ПЛИС XC7Z020-CLG484-1 фирмы Xilinx. Также в данной работе используется микроконтроллер Atmega328p-pu фирмы Atmel. Для программирования микроконтроллера Atmega328p-pu используется программатор USBasp. Далее приводится описание всех используемых в работе устройств.

1.1 Микросхема ПЛИС XC7Z020-CLG484-1

Микросхема ПЛИС XC7Z020-CLG484-1 принадлежит семейству Zynq-7000. Данная микросхема совмещает в себе процессорную систему с двухъядерным ARM Cortex-A9 и 28 нм программируемую логику. Базируется на архитектуре Xilinx программируемой системе на одном кристалле (Xilinx All programmable SoC). [6][3]

Процессорная систему содержит двухъядерный процессор ARM Cortex-A9, работающий на тактовой частоте 800 МГц, имеет прямой доступ памяти (8 каналов) и набор периферии, также имеет двухуровневый кэш: первый уровень – 32 Кбайт инструкций, 32 Кбайт данных на каждое ядро; второй уровень – 512 Кбайт. Процессорная система имеет накристальную память 256 Кбайт, поддерживает оперативную память: DDR2, DDR3, LPDDR2. К набору периферии относятся: 2 USB 2.0 (OTG), 2 трехрежимный Ethernet, 2 SD/SDIO, 2 UART, 2 CAN 2.0B, 2 I2C, 2 SPI, 32 бита GPIO. [6]

Программируемая логика данной микросхемы содержит: 85 тысяч программируемых логических ячеек, 140 блоков памяти по 36 Кбайт, 220 секций DSP, 2 аналогово-цифровых преобразователей по 12 бит, 17 дифференциальных каналов, 195 блоков 3.3 вольтовых ввода/вывода. [6][3]

Взаимодействие между программируемой логикой и процессорной системой осуществляется посредством внутрисистемной шины AXI. [6][7]

1.2 Отладочная плата ZedBoard

ZedBoard является отладочной платой фирмы DIGILENT, основой которой является микросхема ПЛИС XC7Z020-CLG484-1 фирмы Xilinx, принадлежащая семейству Zynq-7000 (Рисунок 1). На отладочной плате расположена оперативная память на 512 МБ DDR3. Отладочная плата имеет следующий набор периферии: разъем JTAG для подключения внешнего загрузочного кабеля; разъемы расширения (FMC, Pmod, XADC); 10/100/1000 Мбит Ethernet; 3.5мм Аудио входы/выходы; USB-OTG 2.0; UART-USB; 1080p HDMI разъём; VGA разъём (12-битный RGB); 128x32 OLED дисплей; 8 светодиодов; 7 тактовых кнопок; 8 переключателей (типа SS-8). Также отладочная плата оснащена встроенным USB-JTAG программатором, что в связке с UART-USB облегчает процесс разработки на данной отладочной плате. Данная плата поддерживает конфигурирование ПЛИС с использованием загрузчика, записанного на SD карту памяти, слот которой расположен на задней стороне отладочной платы. [4][5][2]

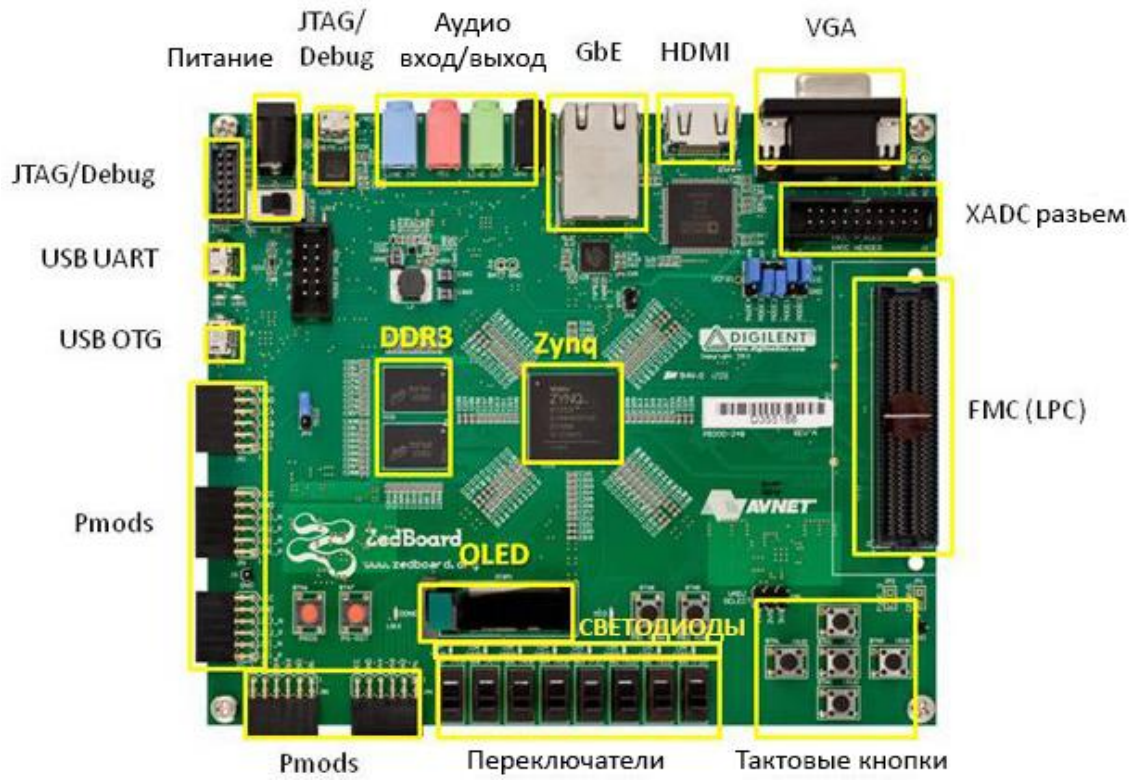


Рисунок 1. Основные элементы отладочной платы ZedBoard

1.3 Микроконтроллер Atmega328p-pu

Atmega328p-pu – это высокопроизводительный маломощный 8-битный CMOS микроконтроллер на базе архитектуры AVR, являющийся расширенной RISC архитектурой. Быстродействие микроконтроллера напрямую зависит от тактовой частоты: 1 MIPS на МГц. Микроконтроллер оснащён: 32 Кбайт ISP флэш-память, 1Кбайт EEPROM память, 2Кбайт SRAM память. Микроконтроллер поддерживает последовательные интерфейсы: I2C, SPI (master/slave), UART. Интерфейс SPI может быть использован для загрузки данных в память микроконтроллера. Также микроконтроллер имеет: два 8-битных таймера, один 16-битный таймер, программируемый порог уровня напряжения питания для запуска микроконтроллера, поддержку прерываний (внутренних/внешних), 23 программируемых выводов, внутренний осциллятор на 8 МГц. Микроконтроллер работает в диапазоне напряжения от 1.8 до 5.5 вольт, потребление в активном режиме 0.2 миллиампер. [9]

1.4 Программатор USBasp

USBasp является USB внутрисхемным программатором для программирования AVR микроконтроллеров фирмы Atmel (Рисунок 2). Основой программатора является микроконтроллер Atmega8. Программатор питается от USB порта. Скорость программирования до 5 Кбайт в секунду. Поддерживает программирование микроконтроллеров с низкой тактовой частотой (<1,5 МГц). Имеет как 5 вольтный режим, так и 3.3 вольтный режим питания программируемого микроконтроллера. Поддерживается операционными системами: Windows, Linux, Mac OS. [10]

Рисунок 2. Внешний вид USBasp

2 Принцип работы и структура разрабатываемой системы «игровая приставка»

В следующих главах произведен краткий обзор основных положений игровой системы, а также приведена структура разрабатываемой системы.

2.1 Основные положения системы «игровая приставка»

Игровая приставка является специализированным электронным устройством, предназначенным для видеоигр. Игровая приставка также относится к сложным системам и состоит из множества подсистем. Основными элементами (подсистемами) игровой приставки являются: центральный процессор, видеоконтроллер, видеопамять и игровой контроллер. Центральный процессор выполняет обработку игровой логики, обработку прерываний с игрового контроллера и запись графического образа в видеопамять. Видеоконтроллер отвечает за своевременное чтение и преобразование графического образа в видеосигнал пригодный для используемого видеоинтерфейса (VGA, HDMI и другие). Игровой контроллер используется как устройство ввода, имеющее микроконтроллер для сканирования нажатий клавиш и отправки собранных данных центральному процессору. Также могут присутствовать другие элементы (подсистемы): звуковой контроллер, сетевой адаптер и другие. [12]

2.2 Структура разрабатываемой системы

Основой разрабатываемой системы «игровая приставка» является отладочная плата ZedBoard. Система включает в себя следующие элементы: центральный процессор, видеоконтроллер, видеопамять, контроллер записи в видеопамять, контроллер ввода/вывода и игровой контроллер (Рисунок 3).

Функцию центрального процессора выполняет ядро ARM Cortex-A9 вычислительной системы, расположенной на микросхеме ПЛИС XC7Z020-CLG484-1. На этой же микросхеме, но уже на основе программируемой логики, выполнен видеоконтроллер, видеопамять, контроллер видеопамяти и контроллер ввода/вывода. Всё общение между вычислительной системой и элементами

программируемой логики осуществляется посредством внутрисистемной AXI-шины.

Видеоинтерфейсом служит разъем VGA, имеющийся на отладочной плате ZedBoard. Использование HDMI разъема является нецелесообразным, так как обладает рядом невостребованных в данной работе функций. [4]

Игровой контроллер на этапе реализации игровой системы выполнен на основе элементов управления, имеющихся на отладочной плате ZedBoard. Позже в работе игровой контроллер реализован в виде отдельного устройства, подключаемого в разъём Pmod отладочной платы, и произведена замена контроллера ввода/вывода на декодер сигнала с игрового контроллера (Рисунок 3).

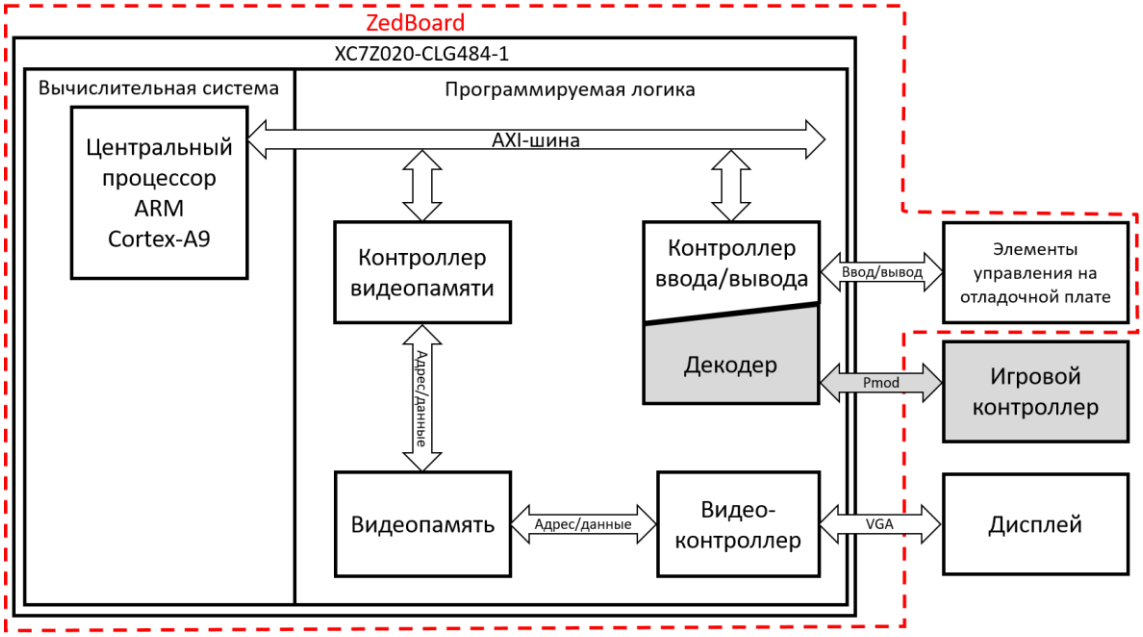


Рисунок 3. Структура разрабатываемой системы на ZedBoard. Контроллер ввода/вывода используется на этапе реализации игровой системы. Интеграция игрового контроллера приводит к замене декодером контроллера ввода/вывода.

3 Реализация игровой системы

В данной главе приведено описание каждого элемента программируемой логики игровой системы на отладочной плате ZedBoard. Также в этой главе описана разработка игры «Змейка» на ядре ARM Cortex-A9 вычислительной системы Zynq-7000. Разработка всей системы происходила в среде разработки Vivado 2015.4 и Xilinx SDK 2015.4.

Система реализована в виде набора IP-блоков (*intellectual property blocks*): блок вычислительной системы ZYNQ7 (ZYNQ7 *Processing System*), блок сброса системы процессора (*Processor System Reset*), блок межсоединения AXI (AXI *Interconnect*), блок ввода/вывода (AXI GPIO), блок видеоконтроллера VGA, три блока памяти (*Block Memodry Generator*) и три блока контроллера памяти. При реализации игровой системы были созданы следующие IP-блоки: контроллер памяти и видеоконтроллера VGA. Остальные IP-блоки были доступны в среде разработки Vivado 2015.4. Система поддерживает вывод изображения разрешением 800 на 600 пикселей с частотой обновления 60 Гц по VGA интерфейсу. Схема соединения IP-блоков разработанной системы изображена в Приложении 2.

IP-блок вычислительной системы ZYNQ7 (ZYNQ7 *Processing System*) выполняет роль центрального процессора игровой системы. Тактовая частота ядер ARM равна 667 МГц. Также данный IP-блок является источником тактирующих импульсов для остальных IP-блоков системы и системной AXI-шины. Для AXI-шины тактовая частота установлена 100 МГц и дополнительно сконфигурирован вывод тактовой частоты 40 МГц для видеоконтроллера VGA. На одно из двух ядер ARM Cortex-A9 написана игра «Змейка» на языке программирования Си. Второе ядро не задействовано. [1] [2]

IP-блоки *Processor System Reset* и AXI *Interconnect* генерируются средой разработки. AXI *Interconnect* организует адресное пространство подключаемых IP-блоков к AXI-шине. *Processor System Reset* отвечает за генерацию сигнала сброса

системы при выполнении внешних или внутренних условиях сброса. Параметры данных двух IP-блоков оставлены по умолчанию. [1] [2]

IP-блок AXI GPIO выполняет роль контроллера ввода/вывода с интерфейсом AXI. Необходим для отслеживания состояния кнопок и переключателей на отладочной плате ZedBoard.

Три IP-блока *Block Memodry Generator* (BRAM) являются видеопамятью игровой системы. Блоки работают в режиме простой двухпортовой памяти (*Simple Dual Port* RAM). Такой режим обеспечивает наличие двух независимых портов: порт чтения, порт записи. Каждый из IP-блоков отвечает за свой цветовой канал (красный, зеленый, синий) и содержит по два бита на пиксель изображения, что в совокупности позволяет выводить изображение в 6-битном RGB режиме и рационально использовать доступное пространство памяти. Порт записи сконфигурирован на запись по 32 бита. Порт чтения – на 2 бита. Исходя из разрешения изображения, адресное пространство на запись равно 30 тысяч адресов, а на чтение 480 тысяч адресов. [2][8]

IP-блок контроллера памяти выполнен на основе пустого IP-блока с AXI4-*Lite* интерфейсом и четырьмя регистрами данного интерфейса, который генерируется при создании нового IP-блока в среде разработки Vivado 2015.4. Контроллер обеспечивает запись в видеопамять через AXI интерфейс. Данный блок выполнен на языке описания аппаратуры VHDL и позволяет записывать 32 бита в память за цикл, что соответствует 16 пикселям изображения. [1][7][16]

IP-блок видеоконтроллера VGA также был выполнен на языке описания аппаратуры VHDL. Обеспечивает своевременное чтение видеопамяти и генерацию выходных сигналов необходимых для корректной работы VGA видеоинтерфейса. Тактируется на частоте 40 МГц, которая является пиксельной частотой для выбранного разрешения. [14]

Данная система может быть легко расширена путем добавления новых IP-блоков или путем изменения уже имеющихся IP-блоков и их настроек. Далее приведено подробное описание созданных IP-блоков игровой системы и разработка игры «Змейка» на ядре ARM Cortex-A9.

3.1 Контроллер памяти

Ядро интерфейса AXI4-*Lite* при генерации нового IP-блока в среде разработки Vivado 2015.4, может иметь минимум 4 регистра по 32 бита. В данном контроллере памяти задействованы все 4 регистра и используются как: регистр адреса, регистр данных и два регистра управления. Первый бит первого регистра управления является флагом готовности данных для записи со стороны ARM. Первый бит второго регистра управления является флагом завершения записи данных в память со стороны контроллера. Остальные биты регистров управления не используются. Такое относительно нерациональное использование регистров вызвано тем что запись в регистр не может быть произведена из разных процессов. Для доступа записи во второй регистр управления со стороны контроллера, в ядре интерфейса AXI4-*Lite* данного IP-блока была отключена возможность записи в данный регистр со стороны AXI интерфейса. [1][7][16]

Алгоритм работы контроллера памяти имеет четыре состояния: бездействие, ожидание, запись и завершение записи (Рисунок 4).

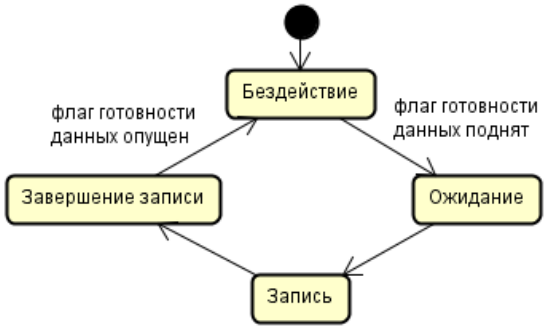


Рисунок 4. Диаграмма состояний контроллера памяти

В состоянии «бездействие» (*idle*) происходит ожидание восходящего фронта флага готовности данных для записи, при этом флаг завершения записи опущен. Если флаг поднят, то на порте записи BRAM выставляется полученные ранее 14 бит адреса и 32 бит данных. Далее разрешается запись путем установки логической единицы на входе wea [0:0] порта BRAM, после чего контроллер памяти переходит в следующее состояние «ожидание».

В состоянии «ожидание» (*waiting*) контроллер пропускает один такт для записи данных в память и переходит в следующее состояние «запись».

В состоянии «записи» (*writing*), контроллер поднимает флаг о завершении записи и запрещается дальнейшая запись путем логического нуля на входе wea [0:0] порта BRAM, после чего переходит в следующее состояние «завершение записи».

В состоянии «завершение записи» (*written*) контроллер ожидает нисходящий фронт флага готовности данных и после переходит в состояние «бездействие».

3.2 Видеоконтроллер VGA

Главной задачей видеоконтроллера является своевременная генерация выходных сигналов необходимых для поддержания вывода изображения по видеоинтерфейсу VGA. Реализованный видеоконтроллер работает на частоте 40 МГц и обеспечивает вывод стабильного изображения разрешением 800 на 600 пикселей с частотой обновления кадра 60 Гц. Данный формат выводимого изображения по VGA видеоинтерфейсу требует выполнения следующих временных норм (Таблица 1). [14]

Таблица 1. Временная характеристика VGA видеоинтерфейса для вывода изображения разрешением 800 на 600 и частотой обновления кадра 60 Гц.

Часть строчной развертки	**Количество пикселей**	**Время (мкс)**
Видимая область	800	20
Передняя площадка	40	1
Синхронизирующий импульс	128	3.2
Задняя площадка	88	2.2
Целая линия	1056	26.4

Кадровая часть	**Количество линий**	**Время (мс)**
Видимая область	600	15.84
Передняя площадка	1	0.0264
Синхронизирующий импульс	4	0.1056
Задняя площадка	23	0.6072
Кадр целиком	628	16.5792

IP-блок видеоконтроллера VGA написан на языке описания аппаратуры VHDL в виде одного процесса. Данный процесс: выполняет чтение цветовых каналов (RGB) из BRAM по 2 бита на канал; выставляет уровень цветовых каналов, считанных из BRAM; своевременно выполняет вертикальную и горизонтальную синхронизацию.

3.3 Разработка игры «Змейка»

Змейка (*Snake*) — компьютерная игра, возникшая в середине или в конце 1970-х. Игрок управляет змейкой, которая ползает по плоскости, собирая еду, а также избегает столкновения с собственным хвостом. Каждый раз, когда змея съедает кусок пищи, она становится длиннее, что постепенно усложняет игру. Игрок управляет направлением движения головы змеи, а хвост змеи движется следом. [13]

Разработка игры «Змейка» под данную игровую систему происходила в среде разработки Xilinx SDK 2015.4. Код игры написан на языке программирования Си, содержит набор функций общения с элементами программируемой логики подключенными к AXI-шине и обработку внутриигровых событий. Для реализации обмена данных по AXI-шине использовалась библиотека «xil_io».

Игровое поле, разрешение которого 800 на 600, поделено на квадраты по 16 на 16 пикселей. Такое разделение используется исходя из возможности контроллера памяти записи строки из 16 пикселей в видеопамять за один цикл. В итоге такого разделения игровое поле представляет собой матрицу элементов из 50 столбцов и 35 строк. Строка высотой 40 пикселей в нижней части экрана используется для вывода игровой статистики.

Все изображения объектов хранятся в отдельном заголовочном файле «sprites.h» в виде матриц. Файл содержит три трёхмерных матрицы, которые представляют собой массивы из двухмерных изображений, где цвет каждого пикселя представлен 8-битами в шестнадцатеричной системе счисления (по два бита на цветовой канал: красный, зеленый, синий и два последних бита не используются). Всего содержит 49 изображений разрешением 16 на 16 пикселей. Первый массив содержит набор изображений частей тел змеи и изображение фрукта. Второй массив содержит изображения цифр от 0 до 9. Третий массив содержит буквы необходимые для внутриигровых надписей.

За запись изображений по требуемым координатам в видеопамять отвечает функция «prSprite». Эта функция принимает следующие входные параметры: тип объекта (змея/цифра/буква), порядковый номер спрайта в массиве изображении, координата X, координата Y и цветовая маска (используется для придания цвета цифрам и буквам, по умолчанию цифры и буквы белые). Данная функция разделяет

изображение объекта, хранимое в заголовочном файле «sprites.h», на цветовые каналы и построчно записывает в соответствующие цветовые каналы видеопамяти при помощи функции «writeinReg».

Функция «writeinReg» отвечает за запись данных в цветовые каналы видеопамять через ранее созданный IP-блок контроллера памяти, учитывая поднятие соответствующих флагов. Функция принимает три входных значения: AXI-адрес IP-блока, адрес ячейки видеопамяти, данные.

Тело игрового персонажа (змея) и фрукт выполнены в виде структур с указателями на следующий и предыдущий элемент. При добавлении нового элемента тела используется динамическое выделение памяти при помощи функции «malloc» библиотеки «stdlib». Помимо указателей структура содержит: координаты объекта (x, y), тип объекта (тело, голова, хвост, фрукт), направление объекта (вверх, вправо, вниз, влево).

Игровая логика разделена на условные состояния, где переход из одного в другой происходит при определенных условиях (Рисунок 5). Например, для перехода из начального экрана в экран игры необходимо нажать кнопку «OK».

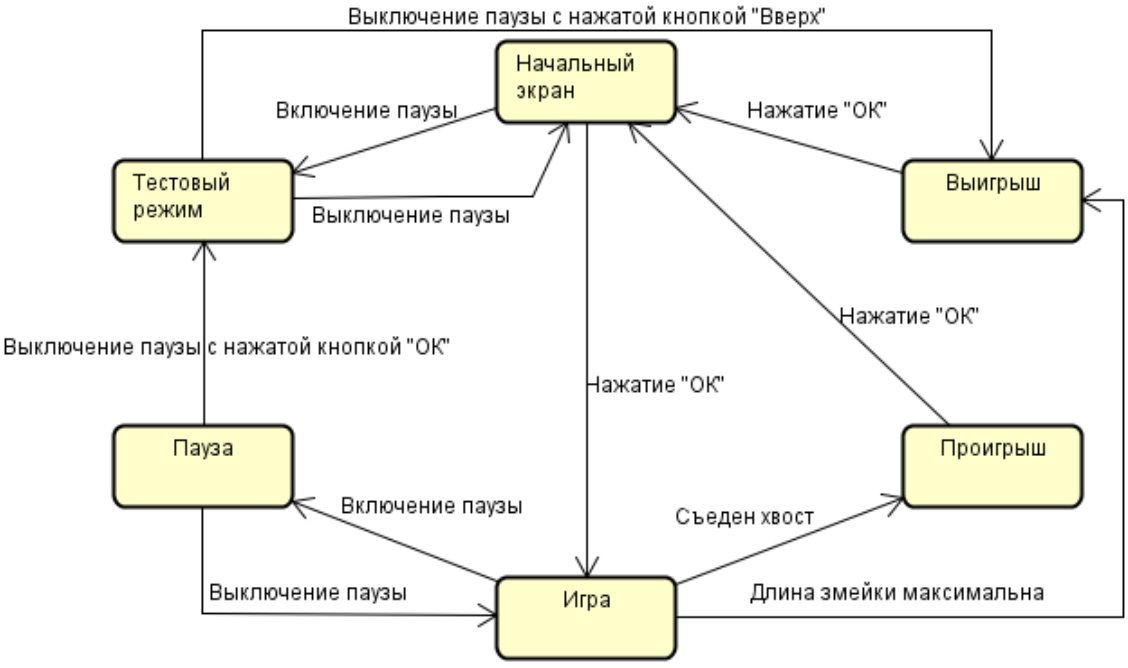


Рисунок 5. Диаграмма состояний игры и условия перехода

Состояние «Начальный экран» подразумевает инициализацию персонажа (змеи), вывод приветствующего сообщения на экран и отсчитывание количества времени нахождения в данном состоянии. Время нахождения в данном состоянии используется для инициализации алгоритма генерации псевдослучайных чисел, который в свою очередь генерирует случайное следующее положения фрукта в ходе игры.

В состоянии «Игра» выполняется проверка нажатий кнопок, задает направление движения персонажа (змеи), выполняется проверка игровых условий, при съедании фрукта начисляются очки и генерируется положение нового фрукта с использованием функции «rand» из библиотеки «stdlib».

В состояниях «Проиграл/Выиграл/Пауза» выполняется остановка хода игры и отображение соответствующих игровых сообщений. При переходе из состояния «Пауза» в состояние «Игра» ход игры возобновляется с того же места, где был прерван.

Состояние «Тестовый экран» используется для отладки. В данном состоянии выводятся все доступные цвета в левой верхней части экрана и выводится график функции «sin(K)» с изменяемым по времени параметром K, который отображен в верхней части экрана.

Также в игру была добавлена возможность регулировки скорости хода игры, путем изменения длительности задержки между ходами персонажа (змеи). При большей скорости, за съедание фрукта начисляется большее количество очков. Элементы управления приведены ниже (Рисунок 6). [4]

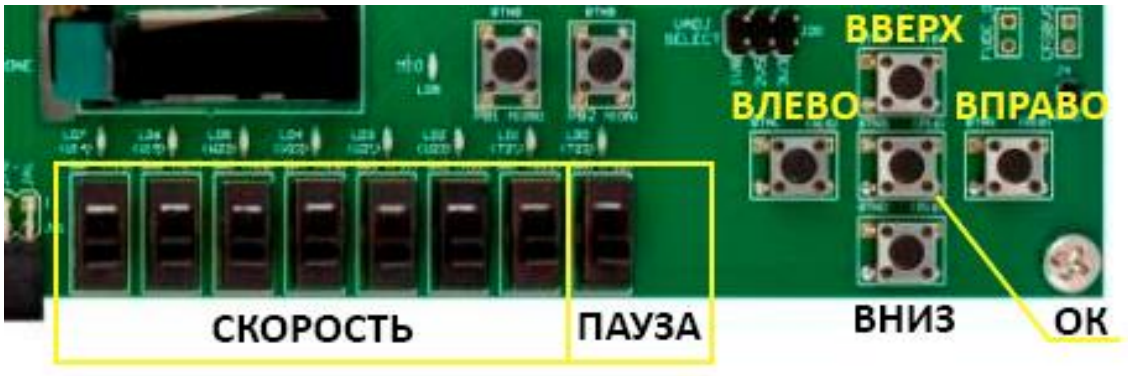


Рисунок 6. Используемые элементы управления отладочной платы ZedBoard

4 Разработка и реализация игрового контроллера

Для расширения игровой системы был разработан и реализован игровой контроллер, в основе которого лежит микроконтроллер Atmega328p-pu фирмы Atmel. Данный игровой контроллер представляет собой удерживаемый двумя руками пульт, выполненный на двухсторонней печатной плате и закрытым с передней и задней стороны органическим стеклом. Размеры игрового контроллера 144 мм в ширину и 72 мм в длину.

На схеме (Рисунок 7) отмечены основные элементы разработанного игрового контроллера.

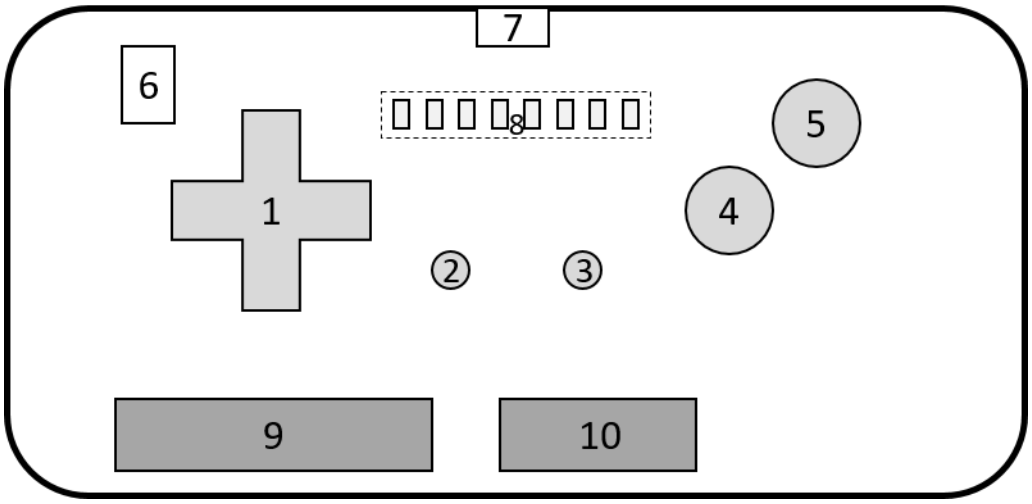


Рисунок 7. Схема основных элементов игрового контроллера: (1-5) элементы управления (кнопки); (6-7) разъемы; (8) ряд из 8 светодиодов; (9) микроконтроллер Atmega328p-pu; (10) микросхема 74HC595.

Из элементов управления игровой контроллер оснащен 8 кнопками (Рисунок 7): (1) четыре кнопки направления (вверх, вправо, вниз, влево) выполненные в виде крестовины, (2) «Пауза», (3) «ОК», (4) «Скорость – » и (5) «Скорость + ».

Из ряда 8 светодиодов: первый (левый) светодиод сигнализирует о работе игрового контроллера, также миганием сигнализирует об активности игрового состояния «Пауза»; остальные 7 светодиодов отображают текущий внутриигрового параметр «Скорость», хранимый в памяти игрового контроллера, от значения 1 (все 7

светодиодов выключены) до значения 8 (все 7 светодиодов активны). Для управления рядом светодиодов используется микросхема 74HC595, являющаяся сдвиговым регистром с последовательным интерфейсом записи. [11]

Для передачи данных, с игрового контроллера на отладочную плату ZedBoard, разработан собственный последовательный однонаправленный интерфейс (Рисунок 7, элемент 7) на базе I2C с использованием 2 линий: линия данных и линия синхронизации. Данный интерфейс подключаться в Pmod порт на отладочной плате ZedBoard. Порт Pmod отладочной платы также является источником питания игрового контроллера (3.3 вольт). Игровой контроллер оснащен портом ICSP (Рисунок 7, элемент 6) для программирования микроконтроллера Atmega328p-pu не извлекая его из печатной платы игрового контроллера. [9]

Микроконтроллер Atmega328p-pu собирает данные о нажатии кнопок, управляет активностью ряда из 8 светодиодов посредством 74HC595 и совершает отправку данных по последовательному интерфейсу на отладочную плату ZedBoard. Atmega328p-pu настроен на работу с использованием внутреннего осциллятора на 8 МГц и напряжением питания начиная с 2.7 вольт. Под данный микроконтроллер было разработано встроенное программное обеспечение на языке Си.

Интеграция игрового контроллера привела к разработке дополнительного IP-блока декодера и замене контроллера ввода/вывода на разработанный блок. Также потребовались небольшие изменения кода игры «Змейка».

Далее приведено подробное описание разработки устройства, настройки работы микроконтроллера, разработки встроенного программного обеспеченья для игрового контроллера и описание интеграции в игровую систему.

4.1 Разработка устройства

Основой игрового контроллера был выбран микроконтроллер Atmega328p-pu фирмы Atmel. Данный микроконтроллер был выбран исходя из его распространенности, надежности работы, низкого энергопотребления и наличие в личном запасе электронных компонентов. Стоит отметить, что данный микроконтроллер может работать в диапазоне от 1.8 до 5.5 вольт и имеет

внутренний осциллятор на 8 МГц, что существенно сокращает количество дополнительных компонентов для обеспечения работы микроконтроллера. [9]

Реализованная ранее игра «Змейка» подразумевает наличие четырех кнопок направления движения змеи, кнопку «ОК», переключатель паузы и переключатели для изменения игрового параметра «Скорость». Исходя из этих требований были реализованы следующие элементы управления: четыре кнопки направления (вверх, вправо, вниз, влево) выполненные в виде крестовины; кнопка «ОК»; кнопка «Пауза» – включает/выключает состояние паузы; кнопки «Скорость – » и «Скорость + » - отвечают за игровой параметр «Скорость». Состояние паузы и текущее значение параметра «Скорость» хранится в памяти микроконтроллера.

Питание игровой контроллер получает через Pmod разъем отладочной платы ZedBoard, которая обеспечивает питание напряжением 3.3 вольта. На линии питания установлен сглаживающий электролитический конденсатор ёмкостью 47 микрофарад. Вывод сброса микроконтроллера подтянут резистором номиналом 100 килоом к положительной линии питания для предотвращения случайного срабатывания. Линия данных и синхронизации, используемые для передачи данных на ZedBoard, подключены к микроконтроллеру через токоограничивающие резисторы номиналом 100 ом. [5]

Поддержка микроконтроллером внутрисхемного последовательного программирования (*In-System Programming, ISP*) позволяет программировать микроконтроллер используя порт ICSP. В противном случае для программирования микроконтроллера требовалось его извлечение из печатной платы. [9]

Для уменьшения количества используемых выводов микроконтроллера использован матричный метод подключения кнопок и использование микросхемы сдвигового регистра 74HC595 с последовательным интерфейсом записи. [11]

Кнопки разделены на два столбца и четыре строки, что сокращает количество необходимых выводом микроконтроллера до 6 (при стандартном подключении кнопок потребовалось бы 8 выводов). Микроконтроллер поочередно подает питание на один из столбцов и проверяет наличие питания на строках, тем самым производится сканирование нажатых кнопок в столбце. Добавлены подтягивающие

к земле резисторы на каждую строку кнопок и диоды последовательно кнопкам. Подтягивающие резисторы номиналом 10 килоом необходимы для гарантии низкого логического уровня на строке при отсутствии питания строки. Диоды необходимы для предотвращения фантомных нажатий. [15]

Микросхема 74HC595, является сдвиговым регистром с последовательным интерфейсом. Для управления рядом из 8 светодиодов данная микросхема используется всего 3 вывода: линия последовательных данных (DS), линия сдвига(SHCP) и линия активации данных (STCP). Дополнительно был добавлен шумоподавляющий керамический конденсатор ёмкостью 0.1 микрофарад на линию активации данных. Выводы сброса (инвертированный MR вывод) для деактивации был подключен к питанию, так как он не востребован. Вывод активации микросхемы (инвертированный OE вывод) на землю, так как предполагается её постоянная активность. [11]

Проектирование электрической схемы, растановка компонентов и последующая трассировка печатной платы производилась в среде Proteus 8 Professional. Принципиальная электрическая схема, выполненная в среде разработки Proteus 8 Professional, доступна для ознакомления в Приложении 1.

Используется двухсторонняя печатная плата, размерами 144 мм в ширину и 72 мм в длину. После расстановки компонентов на печатной плате, была произведенна трасировка в полуавтоматическом режиме (Рисунок 8). На рисунке дорожки на задней стороне платы обозначены синим цветом, дорожки на передней стороне – красным. Зеленой линией обозначено упущенное при проектировании соединение, позже выполненное в виде провода.

Печатная плата была изготовлена в домашних условиях при помощи лазерно-утюжной технологии. Данный метод заключается в распечатки лазерным принтером схемы дорожек в реальном размере на бумаге с глянцевым покрытием и последующим переводом тонера с бумаги на медную поверхность платы путем прогревания утюгом. После чего может быть произведена вытравка платы. Данный метод является весьма дешевым и доступным в домашних условиях.

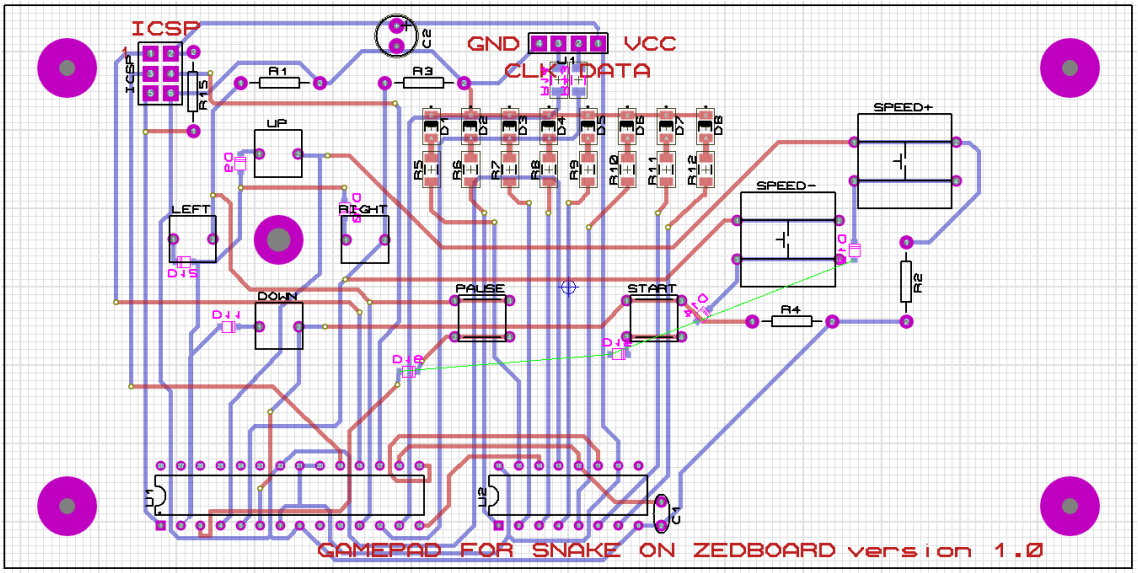


Рисунок 8. Чертеж двухсторонняя печатная плата игрового контроллера в Proteus 8 Professional

После травления платы в растворе персульфата натрия, были просверлены отверстия и распаяны компоненты. Корпус игрового контроллера был выполнен в виде двух пластин из органического стекла толщиной 3 миллиметра, закрывающих печатную плату с передней и задней стороны (Рисунок 9).

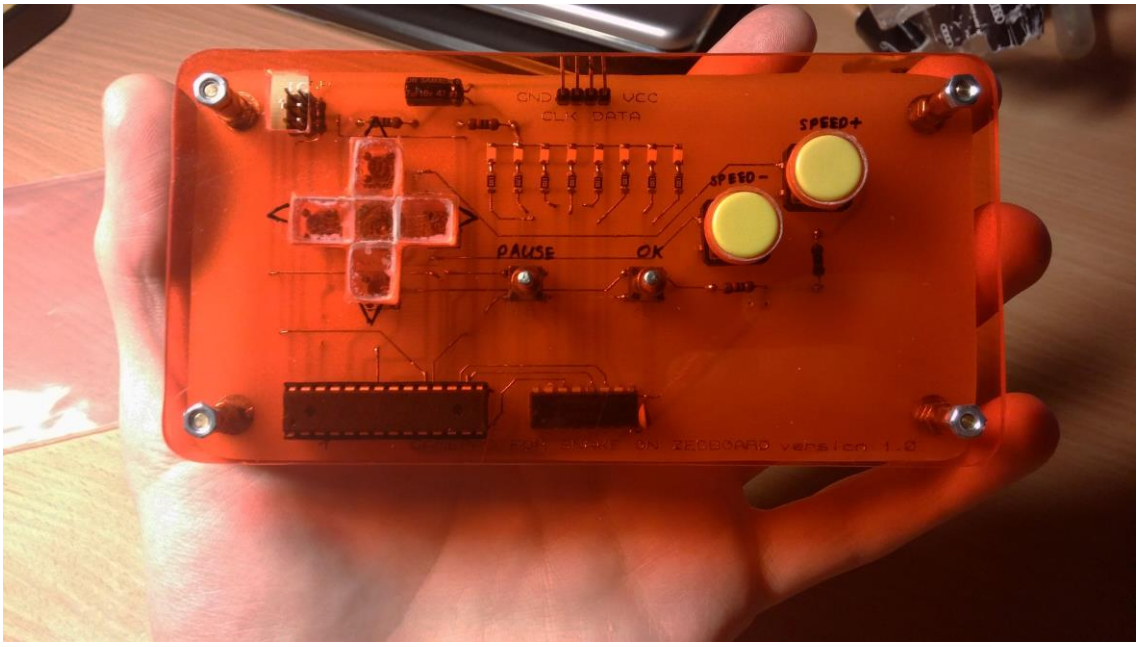


Рисунок 9. Фотография полностью собранного игрового контроллера.

4.2 Настройка режима работы микроконтроллера

Режим работы микроконтроллеров архитектуры AVR задается фьюз битами (*Fuse bits*), которые могут быть изменены только при помощи программатора. Микроконтроллер Atmega328p-pu имеет: младший фьюз байт (*Fuse Low Byte*), старший фьюз байт (*Fuse High Byte*) и расширенный фьюз байт (*Extended Fuse Byte*). Стоит отметить, что в фьюз битах используется негативная логика. [9]

Младший фьюз байт содержит биты определяющие: выбор источника тактов (CKSEL0-3); выбор времени запуска после подачи напряжения или после сигнала сброса (SUT0-1); вывод тактируемой частоты на выводе PORTB0 (CKOUT); деление тактируемой частоты на 8 (CKDIV8). [9]

Старший фьюз байт содержит биты определяющие: выбор вектора сброса (BOOTRST); выбор размера загрузчика (BOOTSZ0-2); запрет на стирания EEPROM память при полном стирании памяти микроконтроллера; контрольный таймер (*Watchdog timer*) всегда активен (WDTON); активация опции debugWIRE; отключение внешнего сброса (RSTDISBL). [9]

Расширенный фьюз байт содержит только три бита задающих минимальный уровень напряжения для запуска микроконтроллера (BODLEVEL0-2). Остальные 5 бит не используются и по умолчанию установлены в «1». [9]

Все фьюз биты оставлены по умолчанию за исключением следующих фьюз битов: биты выбор источника тактов (CKSEL 0-3) сконфигурированы на тактирование микроконтроллера от внутреннего осциллятора с частотой 8 МГц; деление тактируемой частоты на 8 (CKDIV8) выключено для получения максимальной тактовой частоты с внутреннего осциллятора; минимальный уровень напряжения для запуска микроконтроллера (BODLEVEL0-2) установлен 2.7 вольт. Стоит отметить что максимальная тактовая частота работы микроконтроллера ограниченна напряжением питания (Рисунок 10). [9]

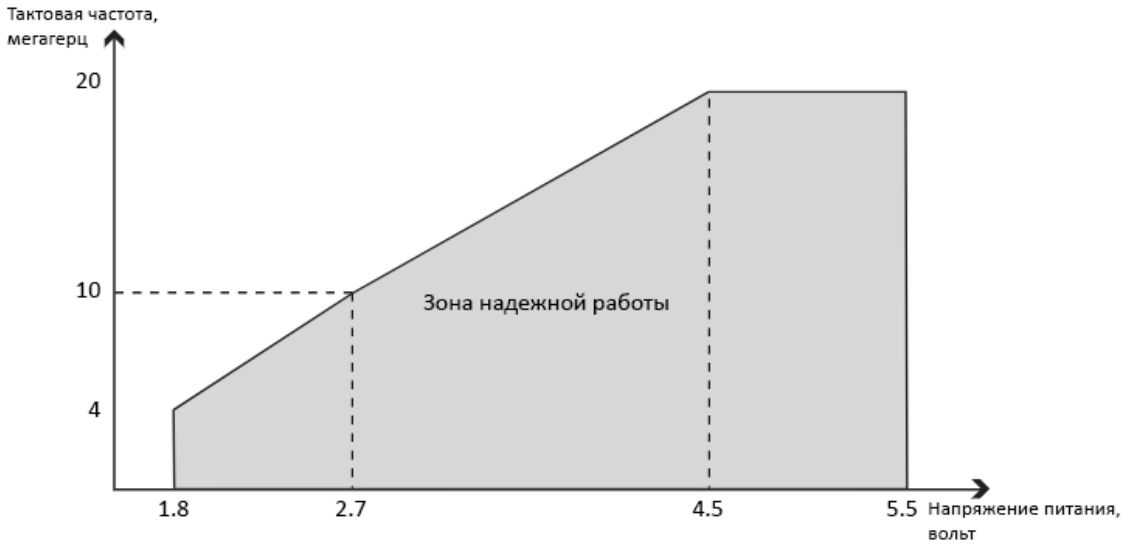


Рисунок 10. График зоны надежной работы.

Исходя из выбранных настроек фьюз биты были установлены в следующее значение (Таблица 2).

Таблица 2. Фьюз (*Fuse*) биты микроконтроллера Atmega328p-pu.

Младший фьюз байт	
CKSEL0	0
CKSEL1	1
CKSEL2	0
CKSEL3	0
SUT0	0
SUT1	1
CKOUT	1
CKDIV8	1

Старший фьюз байт	
BOOTRST	1
BOOTSZ0	0
BOOTSZ1	0
EESAVE	1
WDTON	1
SPIEN	0
DWEN	1
RSTDISBL	1

Расширенный фьюз байт	
BODLEVEL0	1
BODLEVEL1	1
BODLEVEL2	0
-	1
-	1
-	1
-	1
-	1

При помощи программатора USBasp и программы AVRDUDE_PROG 3.3 фьюз биты были успешно загружены в микроконтроллер.

4.3 Разработка встроенного ПО игрового контроллера

Основные задачи встроенного программного обеспеченья (ПО) игрового контроллера: сбор данных; обработка собранных данных; отправка данных центральному процессору игровой системы. Дополнительно игровой контроллер

должен поддерживать индикацию текущего внутриигрового параметра «Скорость» и состояния «Пауза». Встроенное ПО было написана на языке Си в среде разработки Atmel Studio 7.0.

Встроенное ПО начинает работу с основная функции *main* (Рисунок 11). Данная функция состоит из инициализации и бесконечного цикла. При инициализации портов задается их направление (вход/выход) и начальный логический уровень для выводов или подтягивание входов к земле или плюсу питания (*pull-up*/*pull-down*). Инициализация прерываний подразумевает разрешение прерывания по таймеру и предварительную настройку таймера прерывания, который будет использован при отправке данных. В бесконечном цикле происходит сканирование состояния кнопок. Если состояние изменилось с прошлого сканирования, то происходит выполнение проверки нажатия кнопки «Пауза», изменение скорости (если была изменена), отправки данных и сохранение нового состояния кнопок. Далее вне зависимости от состояния кнопок выполняется функция индикации активности паузы и происходит обновление состояния 8 светодиодов.

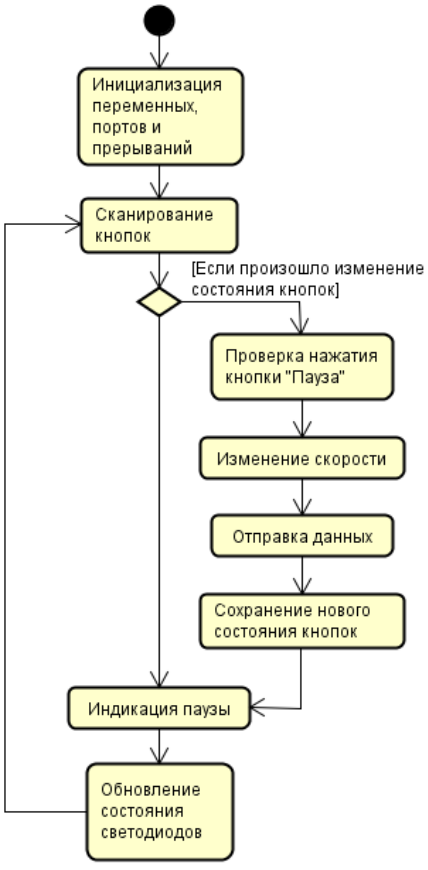


Рисунок 11. Диаграмма деятельности основной функции *main*

Функция сканирования кнопок построена следующим образом (Рисунок 12): инициализация переменных и цикл сканирования. В функции инициализируется две переменные: текущее состояние кнопок и результирующее. Далее в цикле происходит поочередная активация обоих групп кнопок и проверка на активность выводов кнопок. Затем, если пройден первый цикл, то результат сохраняется, если цикл проходит не первый раз, то происходит логическая операция «И» («*AND*») предыдущего результата с текущим результатом проверки. Далее следует задержка в 1мс и цикл повторяется. Эмпирическим путем было выявлено, что для надежного срабатывания достаточно 3 прохода цикла.

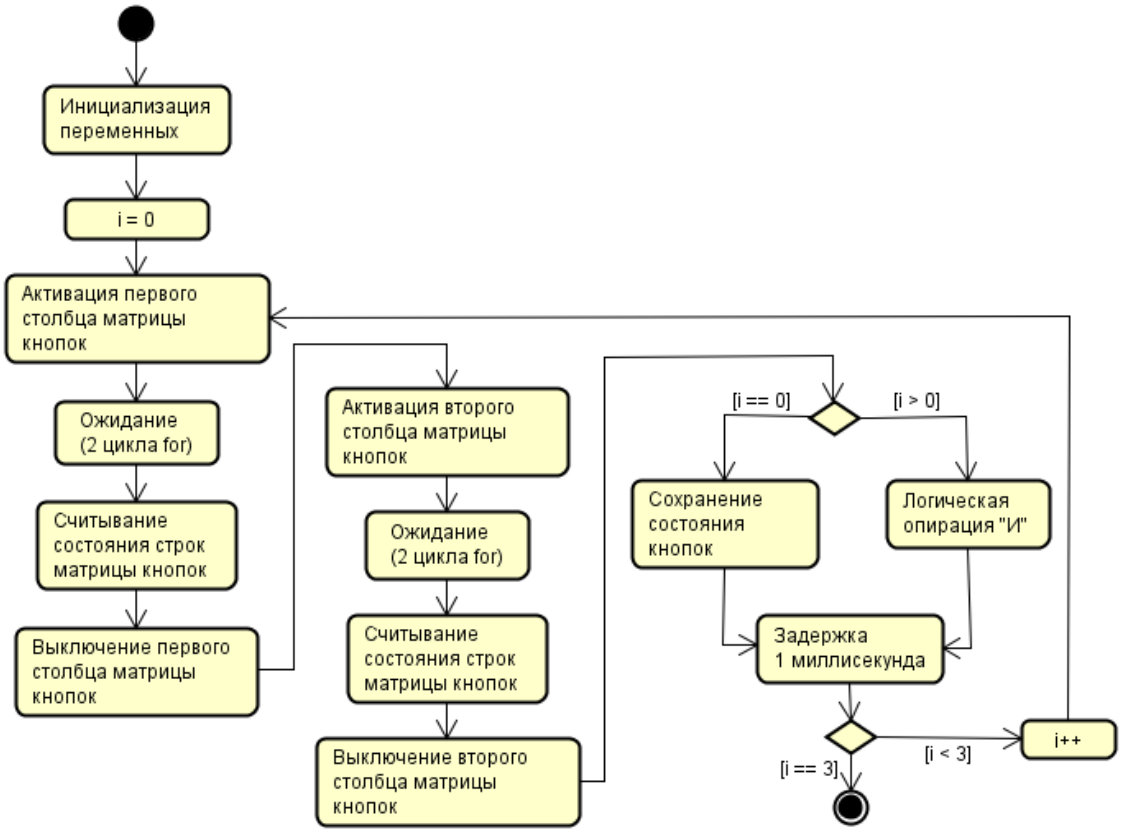


Рисунок 12. Диаграмма деятельности функции сканирования кнопок

Функция изменения состояния паузы является простой логической операцией исключающего «ИЛИ» («*XOR*») текущего состояния кнопки «Пауза» с текущим значением состояния «Пауза».

Функция изменения скорости производит проверку на наличие нажатия на кнопки «Скорость + » и «Скорость – ». При наличии нажатия на соответствующие кнопки происходит либо прибавление значения параметра «Скорость» на единицу (если значение меньше 8), либо уменьшение (если значение больше 0).

В функции отправки данных реализована подготовка и отправка данных с синхронизацией (Рисунок 13). Для передачи данных с фиксированным временем синхронизации используется прерывание по таймеру. Для вызова прерывания используется 8-битный таймер.

В начале функции происходит инициализация переменных. Затем производится преобразование данных в последовательность из 8 бит: первые 3 бита направление в двоичной системе счисления (001 – «вверх», 010 – «вправо», 011 – «вниз», 100 – «влево»); 1 бит значения состояния кнопки «ОК»; 1 бит значения состояния «Пауза»; последние 3 бита значение параметра «Скорость» уменьшенное на единицу (от 0 до 7). Преобразованные данные содержатся во временном буфере для отправки.

Перед циклом отправки происходит активация таймера, путем включения тактовых импульсов для таймера. Внутри цикла отправки происходит выставление логического уровня первого бита буфера отправки на линии данных. Далее происходит уменьшение счетчика на единицу и ожидание поднятия флага разрешения (флаг, поднимаемый прерыванием) на запись следующего бита. После получения разрешения на запись следующего бита, устанавливается низкий логический уровень линии данных, опускается флаг и сдвигается буфер отправки на один бит вправо. Цикл повторяется до конца отправки всех 8 бит. После отправки деактивируется таймер, путем выключения тактирования таймера, и устанавливается низкий логический уровень линии синхронизации. [9]

Прерывание по таймеру представляет собой функцию (Рисунок 13), которая поднимает или опускает фронт на линии синхронизации в зависимости от предыдущего состояния линии синхронизации и при опускании логического уровня на линии синхронизации поднимает флаг разрешения запись следующего бита на линии данных. Прерывание вызывается при достижении счетчика таймера значения 48. Данное значение выбрано исходя из максимального количества тактов необходимых на выполнение сдвига буфера отправки, сброс флага, установки низкого логического уровня линии данных, проверки и установки логического уровня следующего бита из буфера. Таким образом обеспечен одинаковый временной промежуток между сменой уровня линии синхронизации равный 10 микросекундам.

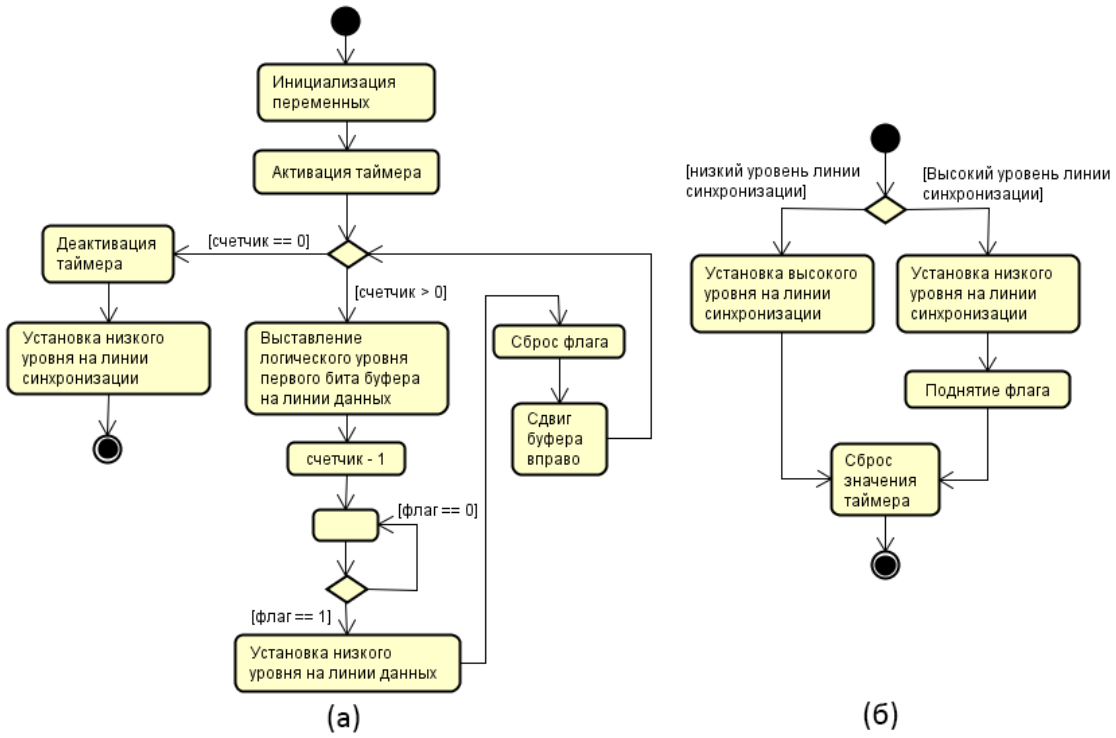


Рисунок 13. Диаграмма деятельности (а) функции отправки данных и (б) функции прерывания

После успешной компиляции, произведенной также в среде разработки Atmel Studio 7.0, при помощи программатора USBasp и программы AVRDUDE_PROG 3.3 скомпилированное встроенное ПО было загружена в флеш-память (*flash memory*) микроконтроллера Atmega328p-pu и занимает 850 байт, что составляет всего 2,6% от всей памяти микроконтроллера, также 11 байт памяти под данных (0,5%).

4.4 Интеграция в игровую систему

Интеграция игрового контроллера в ранее реализованную игровую систему привела к необходимости разработки дополнительного IP-блока декодера сигнала и изменению кода игры «Змейка».

IP-блок AXI GPIO был заменен на IP-блок декодера сигнала. Декодер выполнен на основе IP-блока с интерфейсом AXI4-Lite, сгенерированного средой разработки Vivado 2015.4. Декодер сигнала выполнен языке описания аппаратуры VHDL и содержит два процесса.

Первый процесс тактируется от линии синхронизации. При активации записывает бит с линии данных в регистр и увеличивает счетчик на единицу, который указывает на какой бит регистра записать принятый бит с линии данных. При

достижении счетчиком значения 8, содержимое регистра записывается в первый регистр интерфейса AXI4-Lite. При сбросе процесса обнуляется содержимое регистра и значение счетчика.

Второй процесс тактируется от AXI-шины (100 МГц) и представляет собой таймер ожидания принятия данных, по истечению которого происходит сброс первого процесса. Таймер активируется при обнаружении восходящего фронта сигнала на линии синхронизации. Значение времени ожидания установлено 150 микросекунд.

Дополнительно для отладки регистр принятых данных декодера имеет выводы на 8 светодиодов отладочной платы ZedBoard.

Для поддержки игрой добавленного IP-блока были заменены функции проверки регистров AXI GPIO на функции проверки принятых данных декодером и их преобразование. Дополнительно была изменена функция задержки. В новой функции задержке помимо отсчитывания тактов, производится считывание регистра декодера, что повышает отзывчивость управления в игре.

При помощи осциллоскопа была проверена работа последовательного интерфейса для передачи данных с игрового контроллера в игровую систему (Рисунок 14). Из осциллограммы видно, что данные передаются корректно и сигнал не содержат искажений.

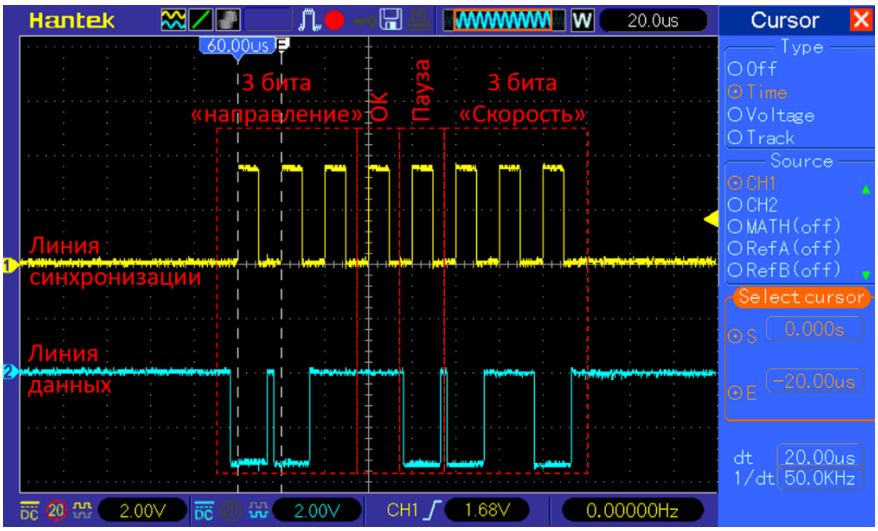


Рисунок 14. Осциллограмма линии синхронизации и линии данных.

Заключение

В данной работе была разработана сложная система с многоуровневой структурой, реализованная на системе на одном кристалле (SoC) семейства Zynq-7000, которая представляет из себя программируемую логическую интегральную схему (ПЛИС) и высокопроизводительный двухъядерный процессор ARM Cortex-A9.

В ходе данной работы были применены полученные навыки программирования, проектирования микросхем на ПЛИС, а также знания в области электроники, компьютерных архитектур и в области дискретной математики, полученные при Таллиннском Техническом Университете по специальности «Вычислительные системы». Также были получены знания в реализации сложных систем. Был изучен принцип работы VGA видеоинтерфейса и AXI-шины, возможности отладочной платы ZedBoard, микроконтроллера Atmega328p-pu, микросхемы ПЛИС XC7Z020-CLG484-1 и их внутренняя структура.

Результатом данной работы является реализованная гибкая игровая система с игрой «Змейка» построенная на отладочной плате ZedBoard, состоящая из множества подсистем: видеопамять, видеоконтроллер VGA, контроллер видеопамяти, центральный процессор и декодер для игрового контроллера. В работе были созданы 3 IP-блока (контроллер памяти, видеоконтроллер VGA, декодер для игрового контроллера), разработана игра «Змейка» для разработанной системы, написано встроенное ПО для игрового контроллера, разработан и реализован игровой контроллер на микроконтроллере Atmega328p-pu. Гибкость системы обеспеченна использованием IP-блоков в роли подсистем. Такой подход обеспечивает возможность расширения данной системы путем добавления новых IP-блоков или путем изменения уже имеющихся IP-блоков.

Данный проект может быть использована в демонстрационных целях возможностей отладочной платы ZedBoard.

Прилагается CD-диск с исходными файлами проекта в Vivado 2015.4 и Atmel Studio 7.0.

Благодарность

Автор выражает благодарность Вацлаву Польскому за оказанную помощь при разработке IP-блоков игровой системы.

Использованная литература

[1] V. Sklyarov, I. Skliarova, J. Silva, A. Rjabov, A. Sudnitson и C. Cardoso, Hardware/Software Co-design for Programmable Systems-on-Chip, TUT PRESS, 2014.

[2] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, Robert W. Stewart, The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC, Strathclyde Academic Media, July 2014.

[3] K. DeHaven, EPPs: The Ideal Solution for a Wide Range of Embedded Systems WP369 (v1.0.1), Xilinx, June 2012.

[4] ZedBoard Hardware User's Guide Rev 2.2, AVNET electronics marketing, January 2014.

[5] ZedBoard Schematics Rev. D.2, Digilent Incorporated, May 2013.

[6] Zynq-7000 All Programmable SoC Overview DS190 (v1.10), Xilinx, September 2016.

[7] AXI Reference Guide UG761 (v13.1), Xilinx, March 2011.

[8] AXI Block RAM (BRAM) Controller v4.0: LogiCORE IP Product Guide, Xilinx, October 2016.

[9] ATmega328/P Complete Datasheet, Atmel Corporation, November 2016.

[10] T. Fischl, «USBasp - USB programmer for Atmel AVR controllers», Fischl [WWW] http://www.fischl.de/usbasp/

[11] SNx4HC595 8-Bit Shift Registers With 3-State Output Registers Datasheet, Texas Instruments, September 2015.

[12] «Video game console», Wikipedia [WWW] https://en.wikipedia.org/wiki/Video_game_console (21.04.2017)

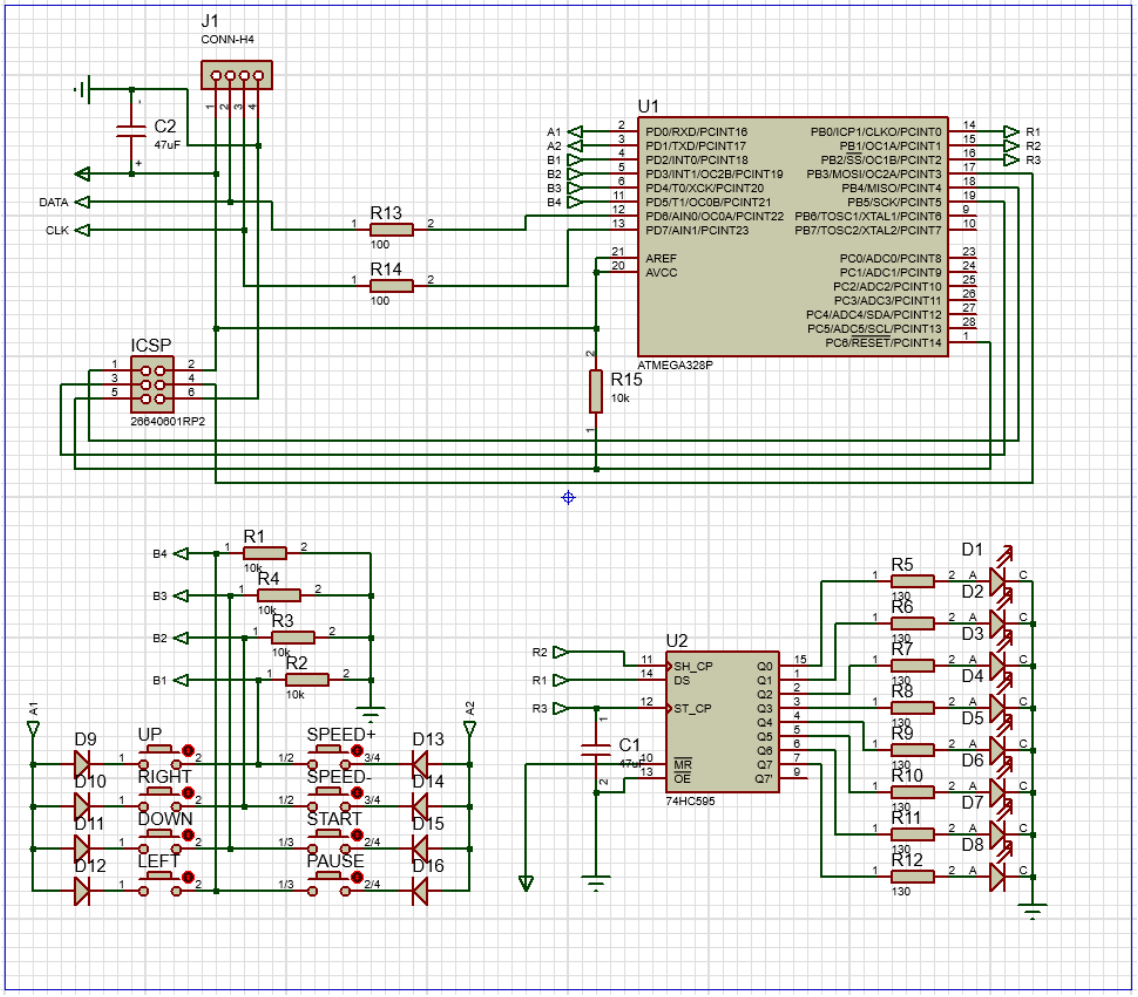
[13] «Snake (игра)», Wikipedia [WWW] https://ru.wikipedia.org/wiki/Snake_(игра) (04.05.2017)

[14] «VGA Signal Timing», TinyVGA [WWW] http://tinyvga.com/vga-timing

[15] firerock, «Как сделать клавиатуру — Матрица», Geektimes [WWW] https://geektimes.ru/post/276358/ (06.03.2017)

[16] «Creating a Custom IP core using the IP Integrator», DIGILENT [WWW] https://reference.digilentinc.com/learn/programmable-logic/tutorials/zybo-creating-custom-ip-cores/start

Приложение 1 – Принципиальная электрическая схема игрового контроллера



Приложение 2 – Схема соединения IP-блоков разработанной системы