

Проектирование контроллера на платформе ПЛИС с помощью нейроинтерфейсов

А. Н. Ильюшина

*Санкт-Петербургский технический колледж
управления и коммерции*

bdbyu@mail.ru

И. М. Новожилов

*Санкт-Петербургский государственный
электротехнический университет
«ЛЭТИ» им. В.И. Ульянова (Ленина)*

novozhilovim@list.ru

Аннотация. В работе рассматривается актуальность выбранной темы, а также приводятся теоретические основы функционирования программируемых логических интегральных схем (ПЛИС) и принцип формирования видеосигнала с использованием интерфейса VGA. Проведён анализ аппаратных и программных средств, применяемых для реализации проекта на выбранной отладочной плате. Выделены ключевые функциональные модули программной части, осуществлена их реализация и верификация. Работоспособность проекта предварительно проверена путём моделирования в программной среде Quartus. После успешного моделирования выполнена реализация проекта на ПЛИС, установленной на отладочной плате. Дополнительно произведён анализ экономической эффективности и себестоимости проекта. В работе также изложены сведения о соблюдении требований охраны труда, приведено пользовательское руководство по эксплуатации разработанной системы. В заключительной части представлены приложения, а также перечень использованных источников. В рамках аппаратной части проекта планируется использование микроконтроллеров семейства Arduino.

Ключевые слова: электричество, реле, напряжение, идентификация

I. ВВЕДЕНИЕ

В настоящее время программируемые логические интегральные схемы (ПЛИС) представляют собой одно из наиболее динамично развивающихся направлений в области цифровой электроники и микроэлектронных технологий. В отличие от традиционных цифровых микросхем, логика функционирования ПЛИС не фиксируется при производстве, а может быть задана пользователем в процессе эксплуатации с применением специализированных средств — аппаратных программаторов и программного обеспечения. Такой подход обеспечивает возможность создания индивидуальных логических структур, обладающих высокой степенью адаптивности и функциональной гибкости.

ПЛИС представляют собой высокоинтегрированные универсальные логические устройства, способные к внутрисистемному перепрограммированию, что делает их особенно актуальными в условиях, требующих оперативной модификации функций аппаратного обеспечения. Актуальность темы исследования обусловлена растущей потребностью в устройствах, обеспечивающих возможность изменения внутренней архитектуры в реальном времени с быстрой перестройкой выполняемых задач.

Особое значение ПЛИС приобретают в условиях интенсивного развития телекоммуникационных технологий, особенно в России, где проектирование специализированной аппаратуры — таких как коммутаторы, модули сопряжения и защиты информации — всё чаще осуществляется на базе ПЛИС. В связи с этим, данные микросхемы активно внедряются в отечественную элементную базу для создания перспективных цифровых систем.

Одной из актуальных проблем, сдерживающих широкое внедрение ПЛИС в образовательный процесс, является недостаток доступных теоретических и практических материалов на русском языке. Решением данной проблемы может стать разработка учебного проекта, позволяющего студентам наглядно ознакомиться с архитектурой и принципами проектирования цифровых устройств на базе ПЛИС. Для повышения наглядности проект включает в себя вывод графической информации на экран посредством интерфейса VGA, а также возможность управления изображением с помощью клавиатуры.

II. РАЗРАБОТКА МАКЕТНОЙ ПЛАТЫ ДЛЯ РАБОТЫ ПЛИС

Отладочная плата QMTECH Intel FPGA Development Board - это инструмент разработки электронных схем, построенный на основе микросхемы ПЛИС Cyclone IV EP4CE15F23C8N. Помимо ПЛИС, отладочная плата содержит две кнопки, два светодиода, семисегментный индикатор с тремя знаками, внешнюю память SDRAM, кварцевый резонатор с частотой 50 МГц, а также интерфейс VGA для вывода изображения, разъёмы Digilent Pmod и 40 контактов для подключения внешних устройств, интерфейс JTAG, предназначенный для коммуникации с компьютером и программирования микросхемы ПЛИС.

Технические характеристики ПЛИС EP4CE15F23C8N:

- Семейство ПЛИС: Cyclone IV;
- Количество логических ячеек: 15408;
- Количество выводов: 484;
- Объём встроенной памяти: 504 Кбит;
- Количество блоков PLL: 4.

Для прошивки ПЛИС используется программатор USB Blaster ALTERA, подключаемый через JTAG интерфейс отладочной платы и USB компьютера. На рисунке была показана отладочная плата QMTECH Intel FPGA Development Board.

Для реализации видеоконтроллера потребуются следующие аппаратные компоненты: внешние устройства (устройства, расположенные на плате) и ресурсы ПЛИС (конкретные порты ввода-вывода).

Для реализации видеоконтроллера были выбраны следующие устройства: семисегментный индикатор, кнопка SW1, разъём VGA и внешняя матричная клавиатура, так как на плате недостаточно кнопок управления. Рассмотрим каждый внешний ресурс станда, который используется в данной работе.

Для опроса клавиатуры необходимо подавать логическую единицу на первый столбец, и считывать поочерёдно с выводов строк значения. После считывания последней строки следует переключиться к следующему столбцу. Если нажать какую-либо из кнопок, то она замкнет вывод строки и вывод столбца, с которыми связана, в результате чего, на соответствующем выводе строки появится логическая единица.

Полная электрическая принципиальная схема представлена ниже.

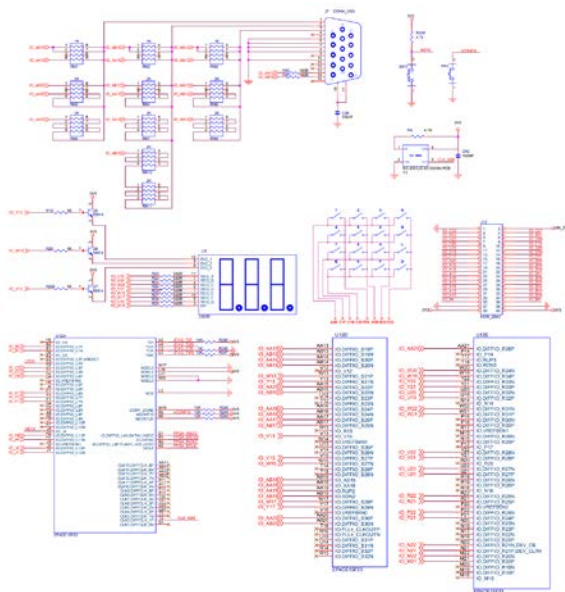


Рис. 1. Полная электрическая схема

III. СОЗДАНИЕ ПРОГРАММНОГО МОДУЛЯ

Для создания VGA видеоконтроллера следует выделить основные подсистемы, из которых он состоит:

Графический контроллер: выполняет роль интерфейса между центральным процессором (CPU) и видеопамятью (VRAM), обеспечивая обмен данными, необходимыми для визуализации. Контроллер обладает функциональностью выполнения битовых операций над передаваемыми данными, что позволяет осуществлять манипуляции с пиксельными данными непосредственно в процессе передачи.

Видеопамять (VRAM): представляет собой буфер хранения данных, отображаемых на мониторе. Организация памяти оптимизирована для хранения пиксельных данных, определяющих цветовую и графическую информацию, подлежащую визуализации.

Последовательный преобразователь (Serializer): осуществляет преобразование данных, хранящихся в видеопамяти в параллельном формате, в последовательный поток битов. Данный поток

передается в контроллер атрибутов для дальнейшей обработки.

Контроллер атрибутов: на основе данных, полученных от последовательного преобразователя, и информации, содержащейся в цветовой палитре (look-up table, LUT), преобразует входные данные в соответствующие цветовые значения (например, RGB). Этот процесс позволяет отображать широкий спектр цветов и оттенков.

Синхронизатор: управляет временными параметрами работы видеоконтроллера, обеспечивая корректную синхронизацию процессов записи и чтения данных из видеопамяти, а также переключение между различными цветовыми слоями (например, для реализации техник двойной буферизации).

Контроллер синхронизации: генерирует синхронизирующие сигналы, соответствующие требованиям стандарта VGA (Video Graphics Array). Эти сигналы обеспечивают правильную развертку изображения на мониторе и синхронизацию между видеоконтроллером и дисплеем.

Вместо реализации обмена данными с процессором, видеопамять для контроллера атрибутов генерируется самостоятельно. Данные, хранящиеся в видеопамяти, представляют собой изображение в виде логотипа «ТКУиК».

Также добавляется модуль управления, с помощью которого можно изменять режим работы и управление видеоизображением через клавиатуру, а для индикации нажатой кнопки и выбранного режима работы используется семисегментный индикатор.

Разработка проектов в среде Quartus Prime допускает реализацию логических схем посредством двух основных методологий:

- Программно-ориентированное проектирование с использованием языков описания аппаратуры (HDL): Этот подход подразумевает кодирование функциональности схемы на одном из специализированных языков, таких как VHDL или Verilog.
- Визуальное (схемотехническое) проектирование: Данный метод основан на графическом построении схемы посредством соединения предварительно определенных логических блоков (стандартных или пользовательских) с использованием визуальных связей (аналог электрических соединений).
- Схемотехническое проектирование обладает следующими преимуществами:
- Интуитивность представления: Визуальная репрезентация схемы способствует более быстрому пониманию ее структуры и функциональности, особенно на начальных этапах разработки.
- Сниженный порог вхождения: Отсутствие необходимости освоения специализированного синтаксиса HDL упрощает процесс проектирования для начинающих разработчиков.

Однако визуальное проектирование имеет и ограничения:

Ограниченная масштабируемость: С ростом сложности схемы экспоненциально увеличивается

количество блоков и связей, что приводит к значительному усложнению схемы, снижению ее читаемости и затруднению процесса отладки.

Сложность повторного использования: Модификация и повторное использование больших схем, созданных визуально, может быть затруднительным.

Несмотря на вышеуказанные ограничения, графическое проектирование не исключается полностью из процесса разработки сложных систем. Оно может быть эффективно использовано для создания отдельных, относительно простых, модулей, которые в дальнейшем интегрируются в общую систему, описанную с использованием HDL.

Для реализации проекта было выбрано сочетание визуального проектирования и языка проектирования аппаратуры в одном проекте. В этом случае отдельные блоки схемы описываются на языке HDL, а главный модуль проекта (Top-Level Entity) объединяет эти блоки в графическом представлении.

После описания входов и выходов они помещаются на главный модуль, представленный на рис. 2.

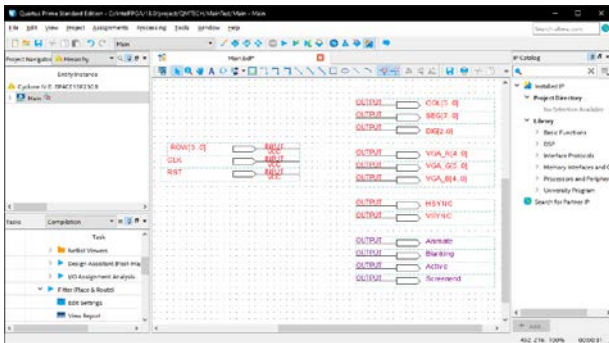


Рис. 2. Главный модуль проекта

Как видно на данном рисунке выходы Animate, Blanking, Active и Screenend выделены другим цветом, так как они не выводятся на плату и кроме вывода Animate используются только для проверки устройства через модуляцию.

Далее рассмотрим проектирование модулей проекта.

Для вывода видеоизображения используется стандартное разрешение VGA экрана 640 по 480 пикселей с частотой 60 Гц. Для правильной настройки сигнала VGA требуется соблюдать частоту 25,175 МГц на пиксель и временные задержки сигнала, представленные в [1-6]. В роли синхронизатора на плате выступает кварцевый резонатор на плате с частотой 50 МГц, но так как эта частота не подходит она изменится при помощи фазовой автоподстройки частоты (ФАПЧ, англ. PLL).

Для создания PLL в среде разработки Quartus используется инструмент Mega Wizard, в котором сначала выбирается нужный тип выходного файла (ALTPLL), а после настраивается. Получение нужной частоты происходит умножением и делением входной частоты на целое число, следовательно, для получения требуемой частоты нужно умножить входную на 1007 и поделить на 2000.

После настройки нужной частоты, рассмотрим модуль контроллера синхронизации.

Для создания модуля синхронизации, необходимо создать новый файл с типом Verilog HDL File. После создания модуля описываются входы и выходы модуля, как представлено в листинге 1.1.

Листинг 1.1 – Описание входов и выходов модуля синхронизации

```
module vga_sync
input wire clk,
input wire rst,
output wire [9:0] x,
output wire [8:0] y,
output wire hsync,
output wire vsync,
output wire animate,
output wire blanking,
output wire active,
output wire screenend
);
```

Далее описываются локальные параметры, при изменении которых можно регулировать разрешение экрана монитора. Список глобальных переменных представлен в [7-12].

Листинг 1.2 – Описание локальных переменных модуля синхронизации.

```
localparam HS_STA = 16;
localparam HS_END = 16 + 96;
localparam HA_STA = 16 + 96 + 48;
localparam VS_STA = 480 + 10;
localparam VS_END = 480 + 10 + 2;
localparam VA_END = 480;
localparam LINE = 800;
localparam SCREEN = 525;
```

Затем описываются локальные регистры, которые в отличие от простых проводов (wire) являются переменными хранения данных. Локальные регистры представлены в листинге 1.3.

Листинг 1.3 – Локальные регистры модуля синхронизации

```
reg [9:0] h_count;
reg [9:0] v_count;
```

После описания локальных параметров и регистров, происходит вычисление выходных данных модуля. Вычисление выходных данных представлено в листинге 1.4.

Листинг 1.4 – Вычисление выходных данных модуля синхронизации.

```
assign hsync = ~((h_count >= HS_STA) & (h_count < HS_END));
assign vsync = ~((v_count >= VS_STA) & (v_count < VS_END));
assign x = (h_count < HA_STA) ? 0 : (h_count - HA_STA);
assign y = (v_count >= VA_END) ? (VA_END - 1) : (v_count);
assign blanking = ((h_count < HA_STA) | (v_count > VA_END - 1));
assign active = ~((h_count < HA_STA) | (v_count > VA_END - 1));
assign screenend = ((v_count == SCREEN - 1) & (h_count == LINE));
```

```
assign animate = ((v_count==VA_END-1)&(h_count==LINE));
```

Для вычисления локальных регистров используется поведенческий блок «always», который срабатывает при положительном фронте синхронимпульса.

Как видно в данном листинге вместе с вычислением регистров также завершается проектирование модуля синхронизации. После описания логики работы модуля создаётся его символьный (визуальный) файл, который помещается на главную схему.

Для удобства модуль управления был разбит на два подмодуля, включающих в себя подмодуль для считывания значения с клавиатуры и подмодуль индикации, обрабатывающий это значение.

Для начала рассмотрим подмодуль клавиатуры. Как и в прошлом модуле создаётся новый файл, и описываются его входы и выходы. Описание входов и выходов подмодуля клавиатуры представлено в [13–16].

Для опроса клавиатуры используется таймер, который увеличивается с каждым фронтом синхронимпульса и при помощи которого последовательно опрашивается каждый столбец клавиатуры. Чтобы не возникало проблем в подмодуле индикации необходимо запретить на клавиатуре нажатие на несколько кнопок. Для этого необходимо использовать временную переменную, которая передаёт значение нажатой кнопки только в конце опроса клавиатуры.

После описания логики работы подмодуля клавиатуры, создаётся его символьный файл, представленный на рис. 3.

Смена режима работы проводится на основе конечного автомата, то есть модели, содержащей конечное число состояний для какого-либо параметра. Блок смены режима, представлен в листинге 1.5.

Листинг 1.5 – Блок смены режима подмодуля индикации.

```
always @ (posedge clk or negedge rst)
begin
if (~rst)
begin
mode_timer <= 0;
mode <= 0;
end
else
begin
case (mode)
0: // автоматический режим работы
if (count == 4'he) mode <= 1;
else
begin
mode <= 0;
value_1 <= 4'hd;
value_2 <= 4'ha;
end
1: // смена режима
if (count == 4'he) mode <= 1;
else
begin
if (mode_timer == 1_000_000)
begin
```

```
mode_timer <= 0;
mode <= 2;
end
else mode_timer <= mode_timer + 1'b1;
end
2: // ручной режим работы
if (count == 4'he) mode <= 3;
else
begin
mode <= 2;
value_1 <= count;
value_2 <= 4'hb;
end
3: // смена режима
if (count == 4'he) mode <= 3;
else
begin
if (mode_timer == 1_000_000)
begin
mode_timer <= 0;
mode <= 0;
end
else mode_timer <= mode_timer + 1'b1;
end
endcase
end
end
```

Как видно в данном листинге присутствует сброс и 4 режима конечного автомата: автоматический (элемент изображения движется самостоятельно), ручной (элемент изображения движется от нажатой кнопки на клавиатуре) и два для перехода на другой режим работы устройства. Также присутствует таймер задержки переключения режима, чтобы устранить случайное переключение от дребезга кнопки.

Далее описывается блок динамической индикации. Смена разрядов индикатора происходит при переполнении таймера смены режима. Время до переполнения таймера подобрано так, чтобы не было заметно переключение между разрядами. Блок индикации представлен в листинге 1.6.

Листинг 1.6 – Блок динамической индикации подмодуля индикации

```
always @ (posedge clk)
begin
if (sw_timer == 0) sw = ~sw;
else if (sw == 1)
begin
value <= value_1;
dig <= 3'b001;
end
else if (sw == 0)
begin
value <= value_2;
dig <= 3'b010;
end
sw_timer <= sw_timer + 1'b1;
```

Отображение на индикатор, происходит преобразованием значения, полученного из предыдущих блоков в значение для сегментов индикатора.

Для увеличения наглядности схемы главного модуля, подмодули помещаются на отдельную схему модуля управления, представленную на рис. 3.

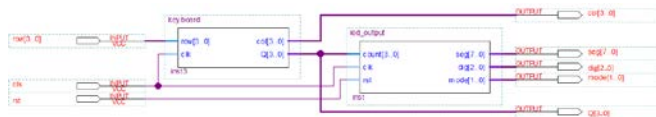


Рис. 3. Схема модуля управления

После создания схемы модуля управления, создаётся его символьный файл, представленный на рис. 4.

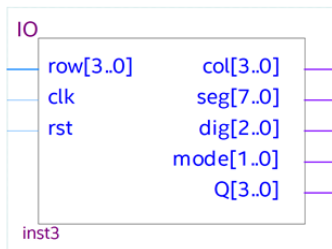


Рис. 4. Символьный файл модуля управления

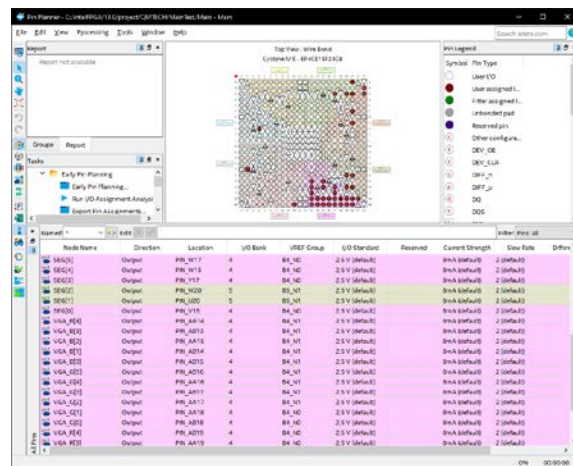
Перед созданием модуля вывода изображения, создаётся файл содержащий данные, отображаемые на экране монитора (видеопамять). После создания файла видеопамяти описываются его входы и выходы, представленные в листинге 1.7.

Листинг 1.7. – Описание входов и выходов видеопамяти

```
module sprite_bitmap
(
    input wire sprite,
    input wire [5:0] yofs,
    output wire [63:0] bits
);
```

Далее описывается локальный регистр, который содержит данные об элементах изображения (спрайтах).

В начале синтеза проводится анализ входов и выходов на схеме и синтаксическая проверка проекта. Как видно из данного рисунка, анализ не выявил ошибок в проекте, но вывел 135 предупреждения. На предупреждения можно не обращать внимания, так как они связаны с вычислением локальных параметров разной разрядности. После анализа проекта необходимо привязать входы и выходы схемы к конкретным выводам ПЛИС. Настройка выполняется с помощью инструмента «Pin Planner», представленном на рис. 5, там же можно настраивать их электрические параметры: стандарт ввода-вывода и максимальный рабочий ток.



Разработанный видеоконтроллер демонстрирует практическую применимость ПЛИС в образовательных целях, и может быть использован в учебных программах по цифровой схемотехнике и встраиваемым системам.

СПИСОК ЛИТЕРАТУРЫ

- [1] Ilyushin Y.V. and Nosova V. A. (2025). Development of Mathematical Model for Forecasting the Production Rate // *International Journal of Engineering*, 38(8), 1749-1757. <https://doi.org/10.5829/ije.2025.38.08b.02>
- [2] Ilyushin Y.V., Boronko E.A. Analysis of Energy Sustainability and Problems of Technological Process of Primary Aluminum Production. // *Energies* 2025, 18, 2194. <https://doi.org/10.3390/en18092194>
- [3] Ilyushin Y., Nosova V., Krauze A. Application of Systems Analysis Methods to Construct a Virtual Model of the Field // *Energies* 2025, 18, 1012. <https://doi.org/10.3390/en18041012>
- [4] Kukharova T., Maltsev P., Novozhilov I. Development of a Control System for Pressure Distribution During Gas Production in a Structurally Complex Field. // *Appl. Syst. Innov.* 2025, 8, 51. <https://doi.org/10.3390/asi8020051>
- [5] Afanaseva O.V., Putilo S.Y., Chirtsov V.V., Demidov A.A. Simulation of the work of structural units of industrial enterprises using the theory of queuing systems // *Academic Journal of Manufacturing Engineering*, 2024, 22(1), 115–126. https://www.ajme.ro/PDF_AJME_2024_1/L13.pdf
- [6] Ereemeeva, A.M., Ilyushin, Y.V. Temperature Control During Storage of Raw Materials in the Process of Biodiesel Fuel Production. *Inventions* 2025, 10, 7. <https://doi.org/10.3390/inventions10010007>
- [7] Kotov, D.D., Pervukhin, D.A., Davardoost, H., Afanasyeva, O.V. (2024). Prospects for the Use of Autonomous Underwater Vehicles (AUV) to Solve the Problems of the Mineral Resources Complex (MRC) of the Russian Federation // *Journal of Maritime Research*, 21(1), 309-317.
- [8] Sidorenko A.A., Dmitriev P.N., Alekseev V.Yu., Sidorenko S.A. Improvement of techno-logical schemes of mining of coal seams prone to spontaneous combustion and rockbumps // *Journal of Mining Institute*. 2023,p.949-961. <https://doi.org/10.31897/PMI.2023.37>