|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра Вычислительной Техники (ВТ)

**КУРСОВАЯ РАБОТА**

по дисциплине «Схемотехника устройств компьютерных систем»

**Тема курсовой работы:** «Пиксеализация изображения»

|  |  |
| --- | --- |
| **Студент группы** ИВБО-04-20 Аксёнов Александр Алексевич  (учебная группа, фамилия, имя, отчество студента) | (подпись студента) |
|  |  |
| **Руководитель курсовой работы**  ассистент кафедры ВТ Люлява Д.В.  (должность, звание, ученая степень) | (подпись руководителя) |
| Работа представлена к защите «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |  |

Допущен к защите «\_\_\_\_\_» \_\_\_\_\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт информационных технологий | |
| Кафедра вычислительной техники | |
|  | **Утверждаю** |
|  | ассистент кафедры ВТ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Люлява Д.В.  *Подпись ФИО* |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022г. |
| **ЗАДАНИЕ** | |
| **на выполнение курсовой работы по дисциплине** | |
| «**Схемотехника устройств компьютерных систем**» | |
|  | |

**Студент:** Свечников Филипп Евгеньевич **Группа:** ИВБО-04-20

# Тема: «Пиксеализация изображения»

**Исходные данные:**

◦ ПЛИС XC7A100TCSG324-1L семейства Artix-7 в составе отладочной платы Xilinx Nexys A7

◦ Клавиатура, работающая по протоколу PS/2

◦ Монитор с интерфейсом, работающим по протоколу VGA.

◦ Маршрут проектирования СБИС

◦ Язык описания аппаратуры Verilog

**Задание:**

1. Ознакомиться с правилами пиксеализации провести анализ предметной области.

2. Разработать набор управляющих модулей согласно правилам игры.

3. Разработать набор модулей, описывающих процессы приёма и обработки данных по протоколу UART.

4. Разработать набор модулей, описывающих драйвер для вывода изображения на монитор по протоколу VGA. 5. Разработать модуль верхнего уровня, реализующий управление для совместной работы всех модулей.

6. Провести верификацию при помощи симуляции.

7. Провести верификацию на отладочной плате.

8. Составить отчетную документацию по проделанной работе.

|  |
| --- |
| **Срок представления к защите курсовой работы: до** **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.** |
|  |
| **Задание на выполнение курсовой работы выдал \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  (Люлява Д.В.)  *Подпись руководителя Ф.И.О. руководителя* |
|  |

**«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.**

**Задание на курсовую работу получил** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Аксёнов А.А.)

*Подпись обучающегося* *Ф.И.О. исполнителя*

**«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.**

### ОТЗЫВ

### на курсовую работу

**по дисциплине**

«Схемотехника устройств компьютерных систем»

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент(ка)** | Аксёнов Александр Алексевич | **Группа** | ИВБО-04-20 |

Характеристика курсовой работы

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Да | Нет | Не полностью |
| 1. Соответствие содержания курсовой работы указанной теме |  |  |  |
| 1. Соответствие курсовой работы заданию |  |  |  |
| 1. Соответствие рекомендациям по оформлению текста, таблиц, рисунков и пр. |  |  |  |
| 1. Полнота выполнения всех пунктов задания |  |  |  |
| 1. Логичность и системность содержания курсовой работы |  |  |  |
| 1. Отсутствие фактических грубых ошибок |  |  |  |

Рекомендуемая оценка: удовлетворительно, хорошо, отлично

|  |  |  |
| --- | --- | --- |
|  | *Подпись руководителя* | Люлява Д.В.  *(ФИО руководителя)* |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | | |

АННОТАЦИЯ

Согласно теме курсовой работы, была разработана системная модель», на высокоуровневом языке программирования Python.

После прохождения предварительного тестирования, на основе системной модели, были созданы модули соответствующих периферийных устройств на языке описания аппаратуры Verilog и разработан модуль, выполняющий основную логику алгоритма. Все модули были протестированы с использование временной диаграммы, а также был сформирован файл проектных ограничений для загрузки на отладочную плату. Тестирования на отладочной было проведено успешно

Данная работа включает в себя 9 рисунков, 6 листингов, 8 приложений. Количество страниц в работе — 25.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc148909089)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc148909091)

[1.1 Описание основного алгоритма пиксеализации 7](#_Toc148909092)

[1.2 Описание работы протокола UART 8](#_Toc148909093)

[1.3 Описание работы протокола VGA 9](#_Toc148909094)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 10](#_Toc148909095)

[2.1 Создание системной модели 10](#_Toc148909096)

[2.2 Создание RTL-модели 12](#_Toc148909097)

[2.2.1 Модуль записи 13](#_Toc148909098)

[2.2.2 Модуль памяти RAM 13](#_Toc148909099)

[2.2.4 Модуль верхнего уровня и файл проектных ограничений 14](#_Toc148909100)

[ТЕСТИРОВАНИЕ 14](#_Toc148909101)

[ЗАКЛЮЧЕНИЕ 16](#_Toc148909102)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 17](#_Toc148909103)

[ПРИЛОЖЕНИЯ 18](#_Toc148909104)

[Приложение А1 19](#_Toc148909105)

[Приложение А2 20](#_Toc148909106)

[Приложение А3 20](#_Toc148909107)

[Приложение А4 21](#_Toc148909108)

[Приложение Б1 24](#_Toc148909109)

[Приложение В1 24](#_Toc148909110)

[Приложение Д1 25](#_Toc148909111)

[Приложение Д2 25](#_Toc148909112)

# ВВЕДЕНИЕ

Цифровая обработка изображения – это широкая сфера It-индустрии, в которой используются одни из самых передовых технологий [1]. Первые техники цифровой обработки изображений были разработаны в 1960-х годах в Лаборатории реактивного движения, Массачусетском технологическом институте, Лабораториях Белла, Мэрилендском университете и других исследовательских центрах в качестве приложений для спутниковой фотосъёмки, преобразования к стандартам фототелеграфа, медицинской визуализации, видеотелефонии, распознавания символов и улучшения фотографий. Одним из подвидов цифровой обработки изображения является пиксеализация - существенное уменьшение разрешения изображения или его части. Отличным решением такой задачи может послужить создание графического адаптера на базе программируемой логической интегральной схемы, в виду её гибкости в плане различного уровня дополнительных настроек, высокой скорости работы и низкой задержки между логическими элементами на кристалле. Также есть возможность использовать широкий пласт компонентов так, или иначе расширяющих возможности по взаимодействию с конечным пользователем.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Описание основного алгоритма пиксеализации

Основной алгоритм пиксеализации изображения, состоит в делении на квадраты (блоки, пиксели) нужного размера, в каждом из них определяется основной цвет, после чего выполняется заливка квадрата полностью этим цветом. Для нашего адаптера будет использоваться изображение 64x64. Предполагается, что вовремя пискеализации размер изображения можно изменять, поэтому после проведение пискеализации с использованием текущего разработанного алгоритма, будет представлено изображения 256х256.Суть программы заключается, в разбитие предоставленного изображения формата jpg , c помощью дополнительной программы на Python, на три отдельных текстовых файла (соответствующих красному, синему и зелёному цвету), с дальнейшим математическим преобразованием в расширенную картинку, с выводом на экран. В ходе преобразования, один текущий пиксель дублируется в четыре соседних.

## 1.2 Описание работы протокола UART

UART — Универсальный асинхронный приёмопередатчик, узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству. Метод преобразования хорошо стандартизован и широко применяется в компьютерной технике. Передача данных в UART осуществляется по одному биту за равные промежутки времени. Этот временной промежуток определяется заданной скоростью UART и для конкретного соединения указывается в бодах (что в данном случае соответствует битам в секунду). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод.

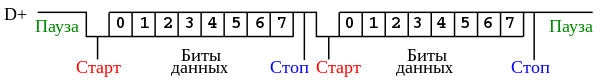
старт-бит — всегда ноль;

8 бит данных;

бит чётности;

стоп-бит — всегда единица.

Для формирования временных интервалов передающий и приёмный UART имеют источник точного времени (тактирования). Точность этого источника должна быть такой, чтобы сумма погрешностей (приёмника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины битового интервала [3]. Детальное описание портов представлено в методических рекомендациях к третьей лабораторной работе [4].

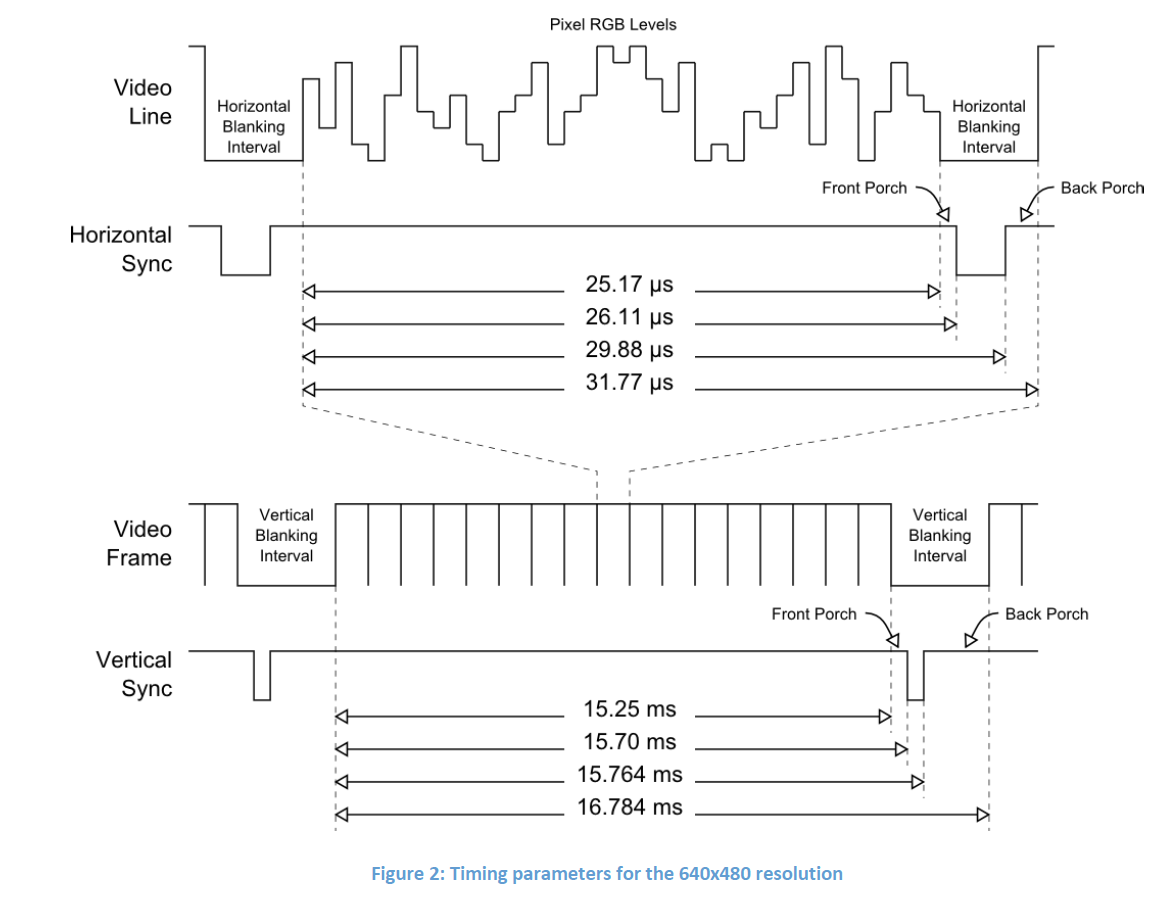


**Рисунок 1.1 – Передача данных по протоколу UART**

## 1.3 Описание работы протокола VGA

**VGA** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Video Graphics Array*) — компонентный видеоинтерфейс, используемый в мониторах и видеоадаптерах. Выпущен IBM в 1987 году для компьютеров PS/2 Model 50 и более старших.

Ключевое отличие VGA от предыдущего видеоадаптера — аналоговый сигнала для передачи цветовой информации. Переход на аналоговый сигнал был обусловлен необходимостью сокращения числа проводов в кабеле. Также аналоговый сигнал давал возможность использовать VGA-мониторы с последующими видеоадаптерами, которые могут выводить большее количество цветов (например, HDMI) [5]. Отображение картинки происходит побитно, с горизонтальной и вертикальной синхронизацией кадра. Более детальное описание портов и соответствующих временных показателей для каждого разрешения представлено в четвертой лабораторной работе [6].

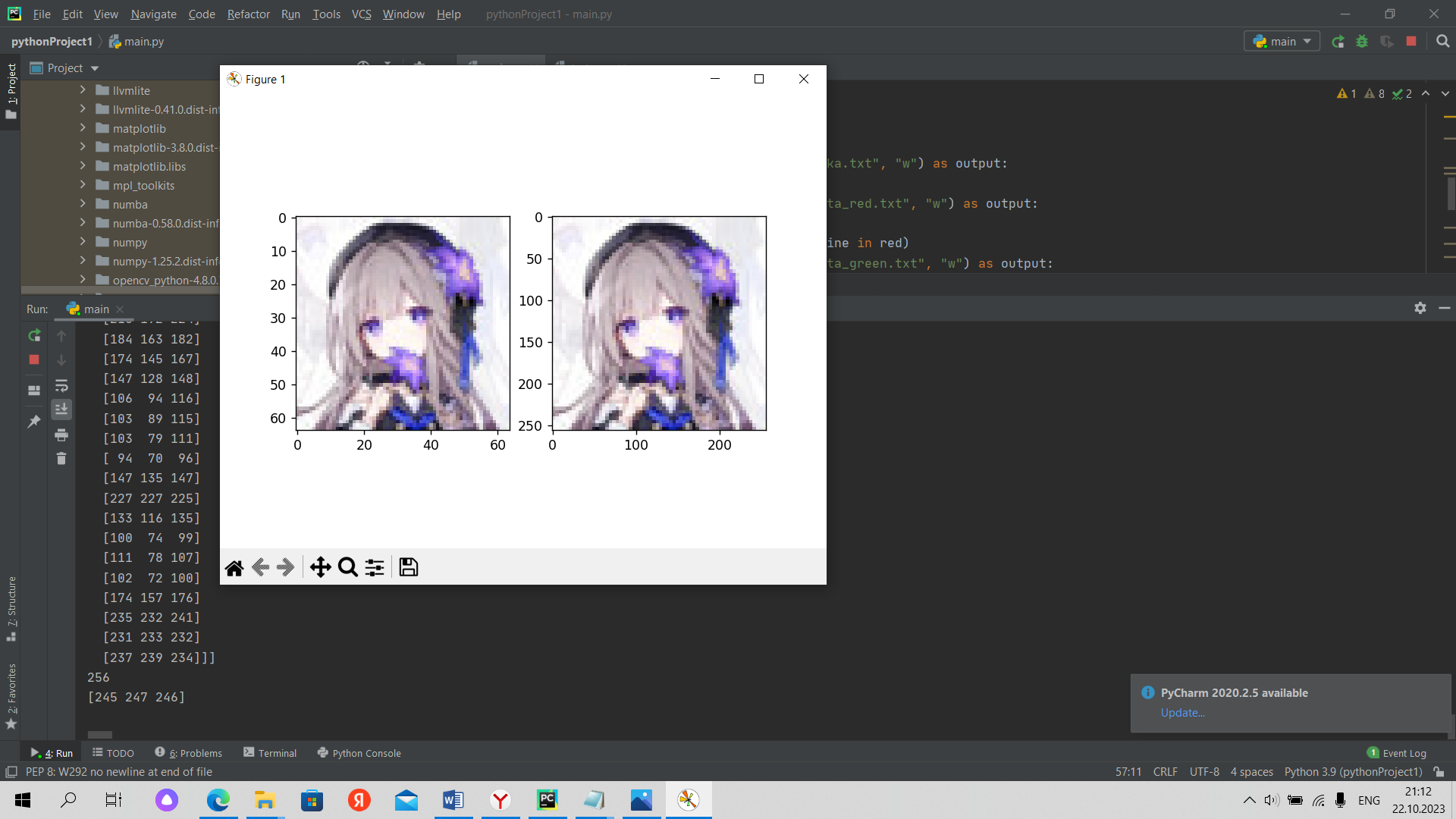


**Рисунок 1.2 – Временная диаграмма VGA**

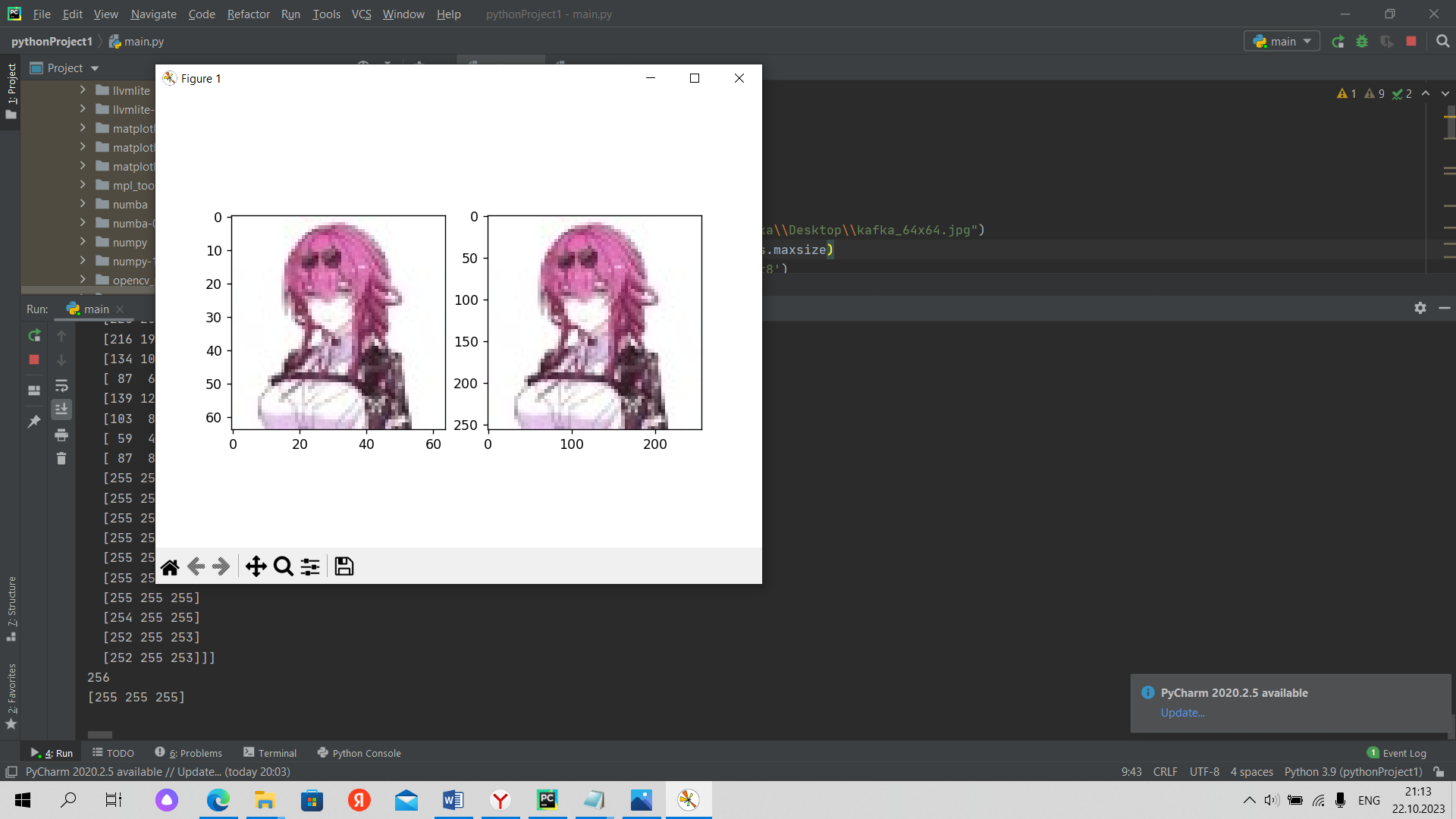
# 2 ПракТИЧЕСКАЯ ЧАСТЬ

## 2.1 Создание системной модели

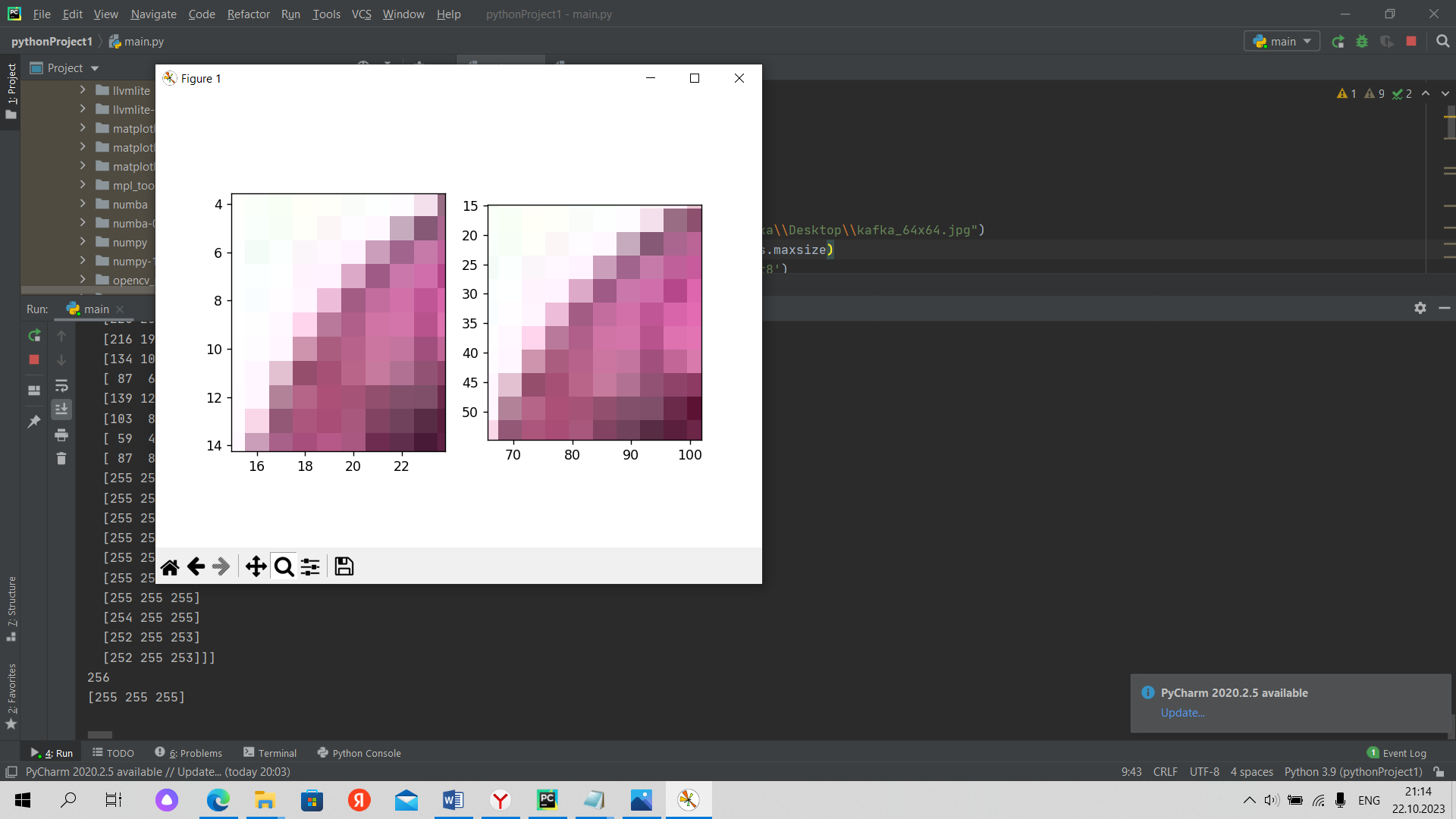
Для проверки основного алгоритма лучшем решением будет представление системной модели в виде программы, написанной на языке программирования высокого уровня [7]. В качестве инструмента был выбран язык python, в следствии удобного взаимодействия с графической библиотекой и удобной реализацией объектно-ориентированного программирования. В основе программы лежит считывание картинки формат jpg, в numpy массив и создание трёх отдельных массивов под каждый цвет, с дальнейшей записью уже преобразованных значений (из int в 16-ричный формат verilog) в отдельные текстовые файлы. Эти значения будут использоваться для загрузки в память ПЛИС. После преобразований и записи в файлы осуществляется отрисовка стартогового изображения и создание его увеличенного варианта, как целой части от деления на 4 (индекс расширения или пиксеализации изображения).



**Рисунок 2.1 – Демонстрация отрисовки изображения 1**



**Рисунок 2.2 – Демонстрация отрисовки изображения 2**

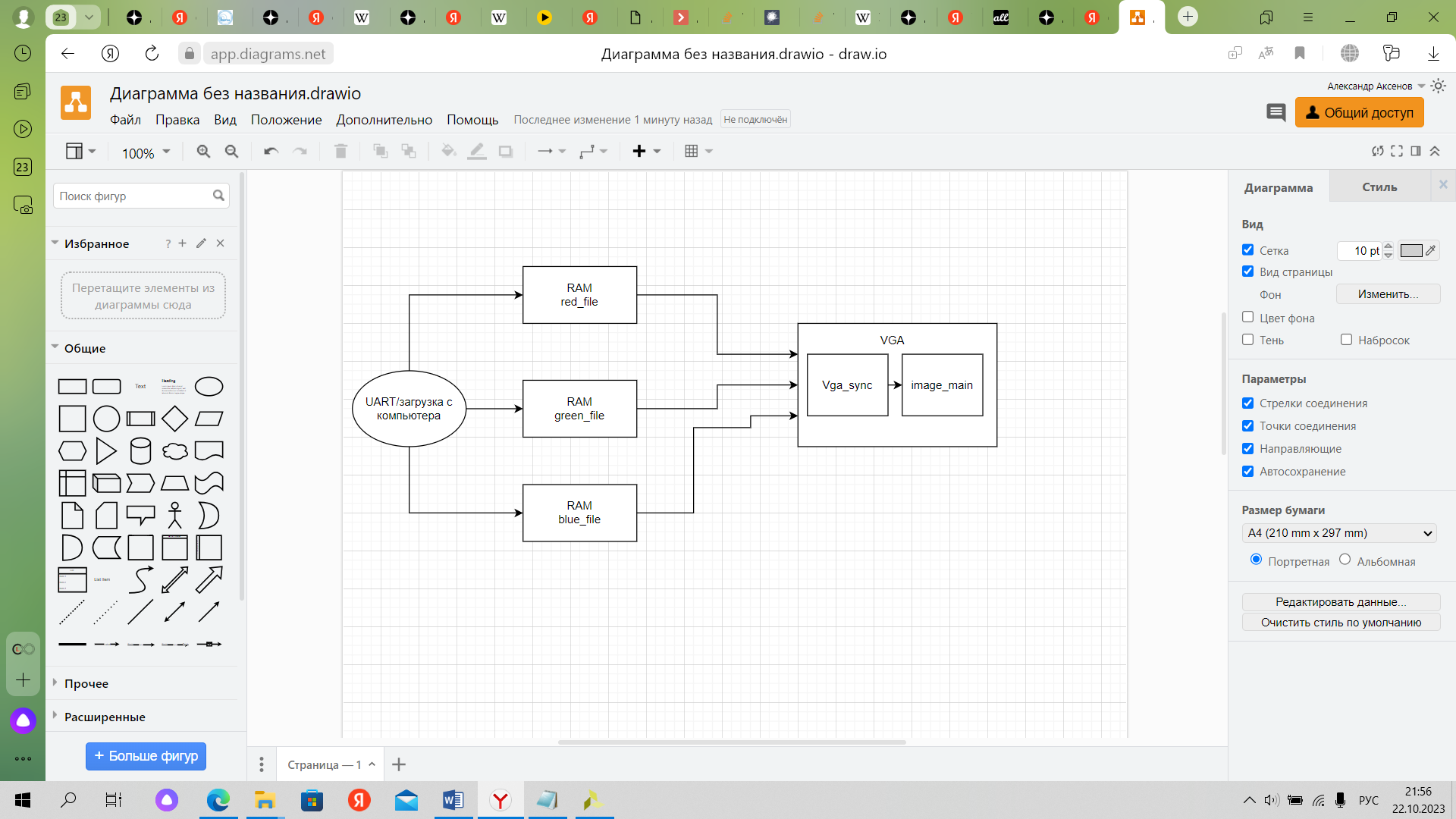
****

**Рисунок 2.2 –Приближение изображения 2**

Итоговая системная модель показывает свою жизнеспособность, существуют некоторые технические особенности реализации, например, отрисовка изображения на основе трехмерного массива с цветами, а не трёх одномерных массивов под каждый цвет. Но основной поток команд остаётся верным.Листинг системной модели представлен в приложении А.1.

## 2.2 Создание RTL-модели

Итоговое устройство можно разделить на три главных модуля: запись значений через uart, три памяти RAM [8] с логикой игры , VGA обработчик. Всё это будет скомпоновано в модуль верхнего уровня. Итоговое графическое представление устройств показано на рисунке 2.2.



**Рисунок 2.2 – Графическое представление устройства**

После предварительного проектирования устройства в общем плане перейдём к рассмотрению его отдельных составляющих.

### 2.2.1 Модуль записи

Основной код модуля предполагает запись изображения в три памяти через UART, однако в связи с недоработкой конечного модуля UART на уровне RTL, выполняется прямая запись в память через копирование значений из текстовых файлов.

### 2.2.2 Модуль памяти RAM

Модуль памяти RAM представляет собой массив из 4096 ячеек. Текущий вид памяти предполагает только режим чтения значений, по заданному адрессу. В дальнейшим при интеграции дополнительного модуля UART предполагается добавить режим записи. Листинг модуля представлен в приложении А.3.

**2.2.3 Модуль VGA**

Основной код модуля VGA был взят из учебника по проектированию плис [9. ст 335] и доработан, согласно варианту курсовой работы. Он состоит из двух модулей vga\_sync и image\_main. Модуль vga\_sync отвечает за горизонтальную и вертикальную синхронизацию, в нём устанавливаются параметры разращения экрана (640х480) , входящая частота преобразуется в 25.175 мгц с помощью pll. Основная задача модуля вести верный подсчёт пикселей и выдавать вертикальную и горизонтальную синхронизацию между front\_porch и back\_porch. В свою очередь модуль image\_main обеспечивает вывод изображения с помощью считывания текущего значения из трёх RAM

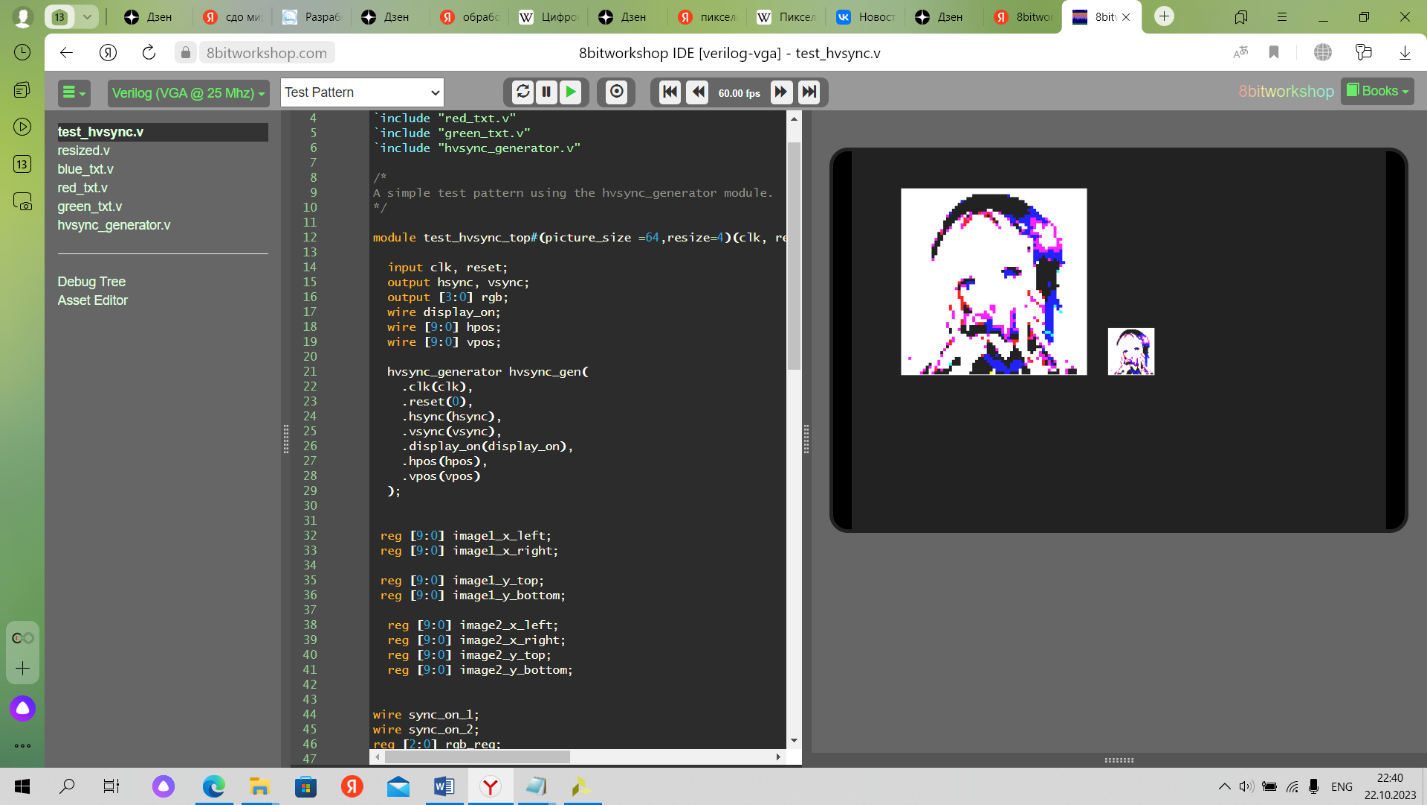
Листинги модулей vga\_sync и image\_main представлен в приложении А.4 и A.5.

### 2.2.4 Модуль верхнего уровня и файл проектных ограничений

В модуле верхнего уровня объединены все модули, перечисленные выше. Также синхросигнал проведён через блок pll во избежание мета стабильности. Листинг файла проектных ограничений представлен в приложении Д.1.

# ТЕСТИРОВАНИЕ

Стоит отметить, что основное тестирование связанно не с системной моделью, а с RTL-реализаций. Предварительные тесты вывода изображения были произведены с помощью платформы 8bitworkshop. На рисунке 3.1 показан вывод изображения 1 и его пиксеализированной копии. Стоит отметить , что в данной среде на каждый цвет представлен всего лишь один бит, поэтому изображение кажется странным. В тесте на плате каждый цвет будет кодироваться 4-мя битами, соответственно изображение будет намного контрастнее.



**Рисунок 3.1 – Вывод изображения 1**

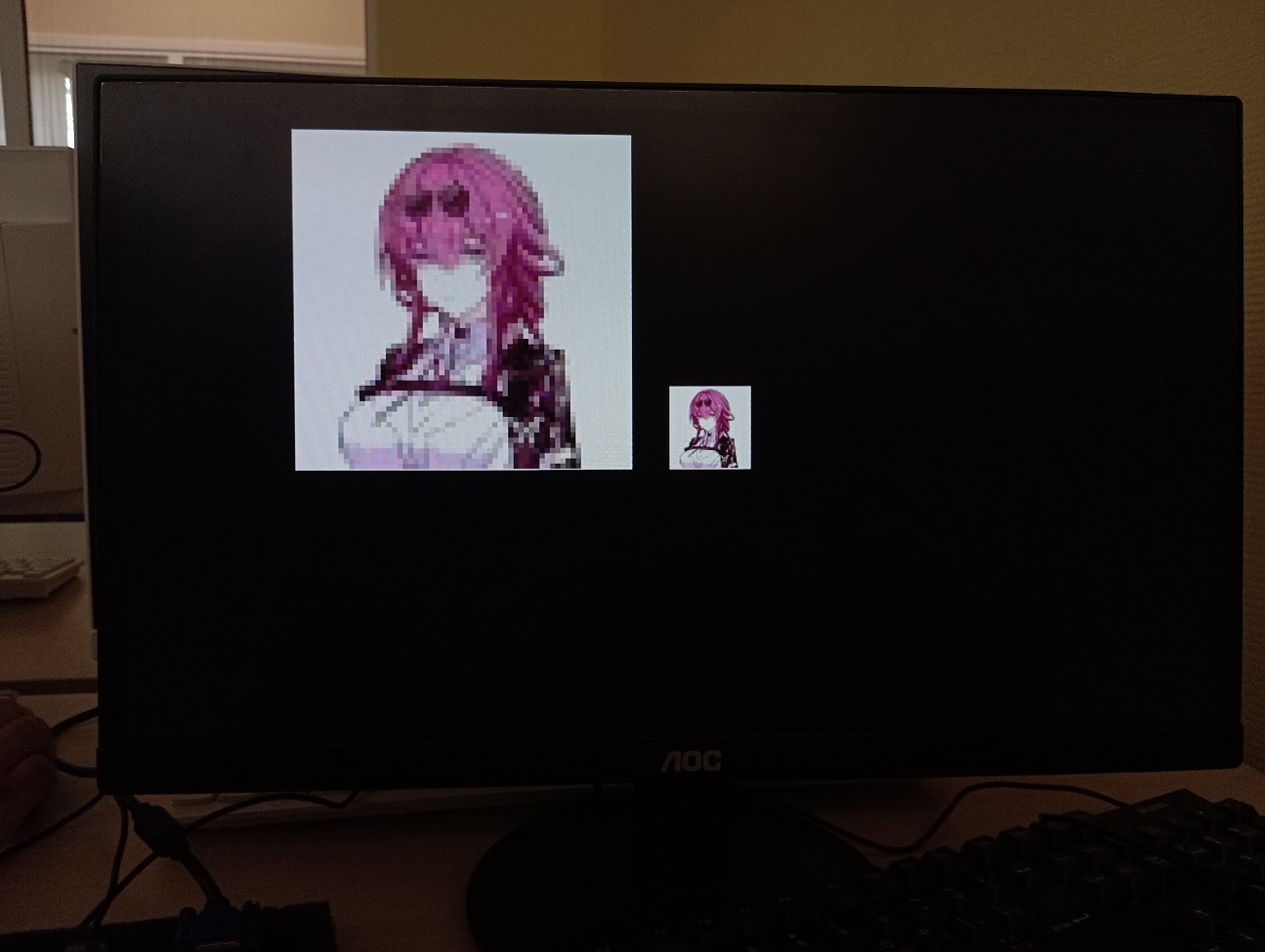
На рисунке 3.2 вывод изображения 1 и его пиксеализированной копии.



**Рисунок 3.2 – Вывод изображения 2**

Дополнительно был протестирован модуль синхронизации VGA Временные диаграммы теста представлены в приложении Б. Листинг теста представлен в приложении В.

Также было произведено финальное тестирование проекта на отладочной плате, в результате которого все представленные модули показали свою корректную работу.

Результат тестирования на отладочной плате показан на рисунке 3.3. 

**Рисунок 3.3– Тестирование на отладочной плате**

# ЗАКЛЮЧЕНИЕ

По итогам проделанной работы был разработан рабочий концепт графического адаптера для пиксеализации изображения, составлены соответствующие модули на языке описания аппаратуры Verilog, однако не удалось реализовать полный функционал. Дальнейшие развития проекта должно включать в себя добавление модуля UART, и реализации возможности записи в память RAM, нового изображения. По взаимодействию с адаптером было создано специальное руководство пользователя, представленное в приложении Д.2.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1.Статья на Википедии про цифровую обработку изображения — URL:

https://ru.wikipedia.org/wiki/Цифровая\_обработка\_изображений

(Дата обращения: 20.10.2023).

2.Статья на Википедии про пиксеализацию изображения — URL:

https://ru.wikipedia.org/wiki/Пикселизация

(Дата обращения: 20.10.2023).

3.Статья на Википедии про интерфейс UART— URL:

https://ru.wikipedia.org/wiki/Универсальный\_асинхронный\_приёмопередатчик

(Дата обращения: 20.10.2023).

4. Методические указания по ЛР № 3 — URL:

https://online-edu.mirea.ru/pluginfile.php?file=%2F969868%2Fmod\_folder%2Fcontent%2F0%2FСправочная%20информация%20к%202-й%20работе.pdf&forcedownload=1 (Дата обращения: 25.12.2022).

5.Статья на Википедии про интерфейс VGA — URL: https://ru.wikipedia.org/wiki/VGA

(Дата обращения: 25.12.2022).

6.Методические указания по ЛР № 4 — URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F969868%2Fmod\_folder%2Fcontent%2F0%2FСправочная%20информация%20к%204-й%20работе.pdf&forcedownload=1

(Дата обращения: 25.12.2022).

7. Тарасов И.Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. — М.: Горячая линия — Телеком, 2021. — 538 с.: ил.

8. Антик М.И. Теория автоматов в проектировании цифровых схем [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА – Российский технологический университет, 2020. — 1 электрон. опт. диск (CD-ROM).

9. Учебник по проектированию ПЛИС — FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version Pong P. Chu published September 2011 528 Pages.

10.Методические указания по ЛР № 1 — URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F969868%2Fmod\_folder%2Fcontent%2F0%2FЛабораторная%20работа%20№%201.pdf&forcedownload=1

# ПРИЛОЖЕНИЯ

Приложение А1 — Листинг системной модели

Приложение A2 — Листинг модуля памяти RAM (без значений)

Приложение A3 — Листинг модуля vga\_sync

Приложение A4 — Листинг модуля Image\_main

Приложение Б1 — Временная диаграмма теста VGA

Приложение В1 — Листинг тест-модуля VGA

Приложение Д1 — Листинг файла проектных ограничений

Приложение Д2 — Руководство пользователя

## Приложение А1

Листинг А1 — Описание системной модели

|  |
| --- |
| import numpy as np  from PIL import Image  import sys  import re  import scipy.misc  import matplotlib.pyplot as plt  Img = Image.open("C:\\Users\\alexa\\Desktop\\kafka\_64x64.jpg")  np.set\_printoptions(threshold=sys.maxsize)  arr = np.asarray(Img, dtype='uint8')  arr = arr.tolist()  arr\_control = np.asarray(Img, dtype='uint8')  #arr\_control = arr\_control.tolist()  red = []  green = []  blue = []  for i in range(len(arr)):  for j in range(len(arr)):  for k in range(3):  buf = str(hex(arr[i][j][k] // 16))  buf = re.sub(r'0x', r"4'h", buf)  buf = buf + ','  arr[i][j][k] = buf  if (k == 0):  red.append(arr[i][j][k])  elif (k == 1):  green.append(arr[i][j][k])  else:  blue.append(arr[i][j][k])  print(red)  print(len(red))  with open("C:\\Users\\alexa\\Desktop\\kafka.txt", "w") as output:  output.write(str(arr))  with open("C:\\Users\\alexa\\Desktop\\kafka\_red.txt", "w") as output:  # output.write(str(red))  output.writelines("%s\n" % line for line in red)  with open("C:\\Users\\alexa\\Desktop\kafka\_green.txt", "w") as output:  # output.write(str(green))  output.writelines("%s\n" % line for line in green)  with open("C:\\Users\\alexa\\Desktop\\kafka\_blue.txt", "w") as output:  # output.write(str(blue))  output.writelines("%s\n" % line for line in blue)  print(arr\_control)  fig, (ax1,ax2,) =plt.subplots(1,2)  arr\_mod = []  for i in range(256):  arr\_mod.append([0] \* 256)  for i in range(256):  for j in range(256):  arr\_mod[i][j] =(arr\_control[i//4][j//4])  print(len(arr\_mod))  print(arr\_mod[0][0])  ax1.imshow(arr\_control, interpolation='nearest')  ax2.imshow(arr\_mod, interpolation='nearest')  plt.show() |

## Приложение А2

*Листинг А2* — *описание* *модуля памяти RAM (без значений)*

|  |
| --- |
| module red\_txt(addr, data);  input [11:0] addr;  output [3:0] data;  assign data = bitarray[addr];  localparam [3:0] bitarray[0:4095] = '{    };  endmodule |

## Приложение А3

*Листинг А3* — *описание модуля vga\_sync*

|  |
| --- |
| `ifndef HVSYNC\_GENERATOR\_H  `define HVSYNC\_GENERATOR\_H  /\*  Video sync generator, used to drive a VGA monitor.  Timing from: https://en.wikipedia.org/wiki/Video\_Graphics\_Array  To use:  - Wire the hsync and vsync signals to top level outputs  - Add a 3-bit (or more) "rgb" output to the top level  \*/  module vga\_sync(clk, reset, hsync, vsync, display\_on, hpos, vpos);  input clk;  input reset;  output reg hsync, vsync;  output display\_on;  output reg [9:0] hpos;  output reg [9:0] vpos;  // declarations for TV-simulator sync parameters  // horizontal constants  parameter H\_DISPLAY = 640; // horizontal display width  parameter H\_BACK = 48; // horizontal left border (back porch)  parameter H\_FRONT = 16; // horizontal right border (front porch)  parameter H\_SYNC = 96; // horizontal sync width  // vertical constants  parameter V\_DISPLAY = 480; // vertical display height  parameter V\_TOP = 33; // vertical top border  parameter V\_BOTTOM = 10; // vertical bottom border  parameter V\_SYNC = 2; // vertical sync # lines  // derived constants  parameter H\_SYNC\_START = H\_DISPLAY + H\_FRONT;  parameter H\_SYNC\_END = H\_DISPLAY + H\_FRONT + H\_SYNC - 1;  parameter H\_MAX = H\_DISPLAY + H\_BACK + H\_FRONT + H\_SYNC - 1;  parameter V\_SYNC\_START = V\_DISPLAY + V\_BOTTOM;  parameter V\_SYNC\_END = V\_DISPLAY + V\_BOTTOM + V\_SYNC - 1;  parameter V\_MAX = V\_DISPLAY + V\_TOP + V\_BOTTOM + V\_SYNC - 1; |

*Продолжение Листинг А3*

|  |
| --- |
| wire hmaxxed = (hpos == H\_MAX) || reset; // set when hpos is maximum  wire vmaxxed = (vpos == V\_MAX) || reset; // set when vpos is maximum  wire clk\_out;  clk\_wiz\_0 m0(.clk\_in1(clk),.clk\_out1(clk\_out),.reset(reset));  // horizontal position counter  always @(posedge clk\_out)  begin  hsync <= ~((hpos>=H\_SYNC\_START && hpos<=H\_SYNC\_END));  if(hmaxxed)  hpos <= 0;  else  hpos <= hpos + 1;  end  // vertical position counter  always @(posedge clk\_out)  begin  vsync <= ~((vpos>=V\_SYNC\_START && vpos<=V\_SYNC\_END));  if(hmaxxed)  if (vmaxxed)  vpos <= 0;  else  vpos <= vpos + 1;  end    // display\_on is set when beam is in "safe" visible frame  assign display\_on = (hpos<H\_DISPLAY) && (vpos<V\_DISPLAY);  endmodule  `endif |

## Приложение А4

*Листинг А4* — *описание модуля image\_main*

|  |
| --- |
| module Image\_main(clk, hsync, vsync, VGA\_R\_3,VGA\_G\_3,VGA\_B\_3,VGA\_R\_2,VGA\_G\_2,VGA\_B\_2,VGA\_R\_1,VGA\_G\_1,VGA\_B\_1,VGA\_R\_0,VGA\_G\_0,VGA\_B\_0);  input clk;  output hsync, vsync;  output VGA\_R\_3,VGA\_G\_3,VGA\_B\_3;  output VGA\_R\_2,VGA\_G\_2,VGA\_B\_2;  output VGA\_R\_1,VGA\_G\_1,VGA\_B\_1;  output VGA\_R\_0,VGA\_G\_0,VGA\_B\_0;    wire display\_on;  wire [9:0] hpos;  wire [9:0] vpos;  reg [2:0] rgb\_reg;  Vga\_sync hvsync\_gen(  .clk(clk),  .hsync(hsync),  .vsync(vsync),  .display\_on(display\_on),  .hpos(hpos),  .vpos(vpos)  ); |

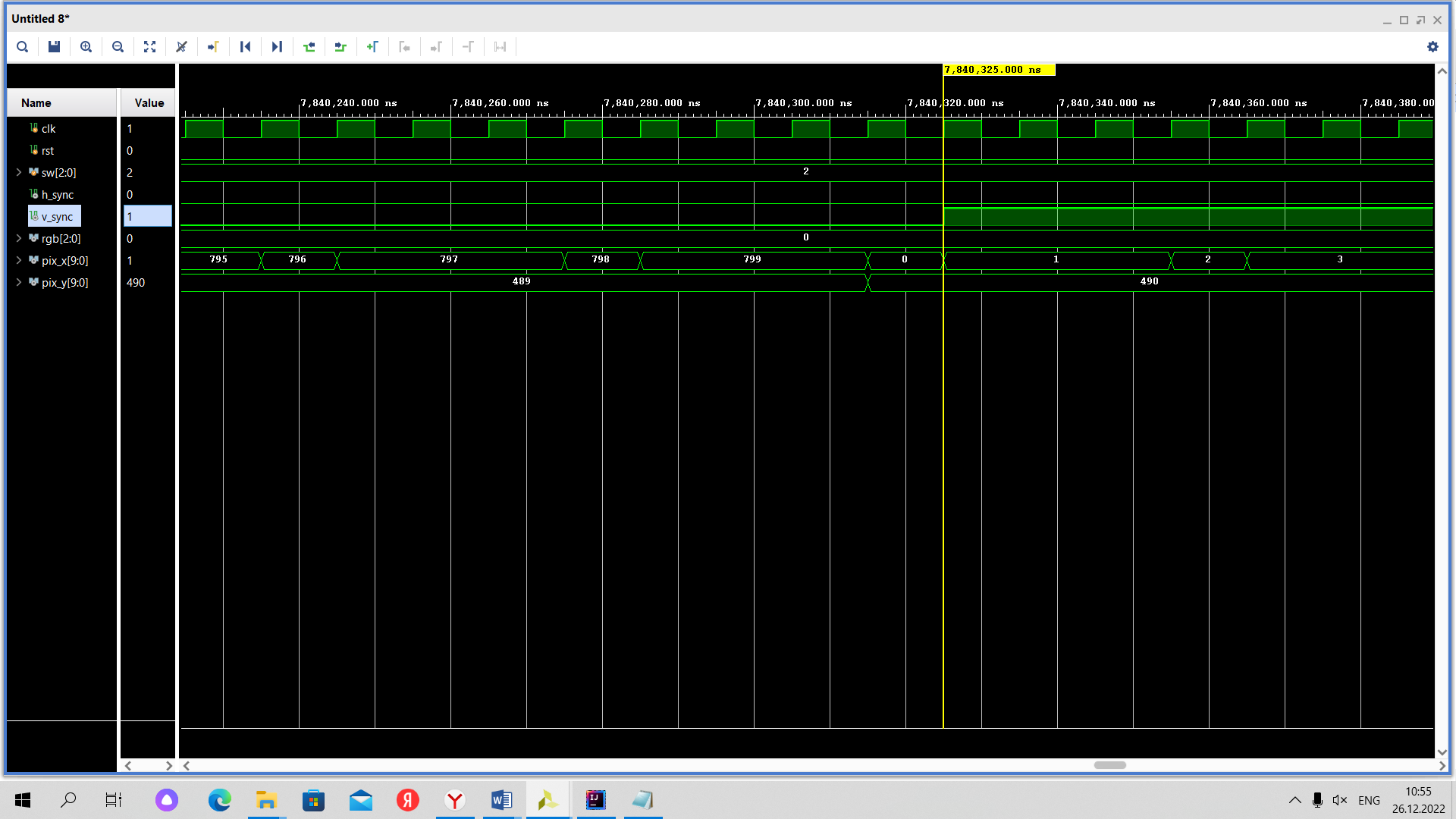
*Продолжение Листинг А4*

|  |
| --- |
| assign VGA\_R\_3 = red\_color[3];  assign VGA\_R\_2 = red\_color[2];  assign VGA\_R\_1 = red\_color[1];  assign VGA\_R\_0 = red\_color[0];  assign VGA\_G\_3 = green\_color[3];  assign VGA\_G\_2 = green\_color[2];  assign VGA\_G\_1 = green\_color[1];  assign VGA\_G\_0 = green\_color[0];  assign VGA\_B\_3 = blue\_color[3];  assign VGA\_B\_2 = blue\_color[2];  assign VGA\_B\_1 = blue\_color[1];  assign VGA\_B\_0 = blue\_color[0];    reg [9:0] image1\_x\_left;  reg [9:0] image1\_x\_right;  reg [9:0] image1\_y\_top;  reg [9:0] image1\_y\_bottom;    reg [9:0] image2\_x\_left;  reg [9:0] image2\_x\_right;  reg [9:0] image2\_y\_top;  reg [9:0] image2\_y\_bottom;  wire sync\_on\_1;  wire sync\_on\_2;  reg [11:0] count;  reg [9:0] scale;  reg[9:0] picture\_size;  reg [9:0] resize\_scale;  reg [11:0] sum\_image1\_1;  reg [11:0] sum\_image1\_2;  reg [11:0] sum\_image2\_1;  reg [11:0] sum\_image2\_2;  wire [3:0] red\_buffer;  wire [3:0] green\_buffer;  wire [3:0] blue\_buffer;  reg [3:0] red\_color;  reg [3:0] blue\_color;  reg [3:0] green\_color;  reg read\_ready;  assign rgb = {1'b0,rgb\_reg};  initial begin  count = 0;  scale =4;  picture\_size =64;  resize\_scale = 1;  sum\_image1\_1=0;  sum\_image1\_2=0;  sum\_image2\_1=0;  sum\_image2\_2=0;  image1\_x\_left = 20;  image1\_x\_right = (picture\_size\*scale)+(image1\_x\_left-1);//209;  image1\_y\_top = 20;  image1\_y\_bottom = (picture\_size\*scale) +image1\_y\_top;  image2\_x\_left = image1\_x\_right +30;  image2\_x\_right = (picture\_size\*resize\_scale)+(image2\_x\_left-1);//209;  image2\_y\_bottom = image1\_y\_bottom;  image2\_y\_top = image2\_y\_bottom-(picture\_size\*resize\_scale);  end |

*Продолжение Листинг А4*

|  |
| --- |
| red\_txt red\_file1 (.addr(count),.data(red\_buffer));  green\_txt green\_file1 (.addr(count),.data(green\_buffer));  blue\_txt blue\_file1 (.addr(count),.data(blue\_buffer));      assign sync\_on\_1 =(image1\_x\_left<=hpos) &&(hpos <=image1\_x\_right) &&(image1\_y\_top <=vpos) && (vpos <=image1\_y\_bottom);  assign sync\_on\_2 =(image2\_x\_left<=hpos) &&(hpos <=image2\_x\_right) &&(image2\_y\_top <=vpos) && (vpos <=image2\_y\_bottom);  always@(posedge clk)begin    if((sync\_on\_1))begin  sum\_image1\_1[9:0] = (hpos-image1\_x\_left)/scale;  sum\_image1\_2[9:0] = (vpos-image1\_y\_top)/scale;  count = (sum\_image1\_1+(picture\_size\*sum\_image1\_2));  end  if((sync\_on\_2))begin  sum\_image2\_1[9:0] = (hpos-image2\_x\_left)/resize\_scale;  sum\_image2\_2[9:0] = (vpos-image2\_y\_top)/resize\_scale;  count = (sum\_image2\_1+(picture\_size\*sum\_image2\_2));  end  end  always@(posedge clk)begin  if((sync\_on\_1||sync\_on\_2)) begin  red\_color = red\_buffer;  blue\_color = blue\_buffer;  green\_color = green\_buffer;  end  else begin  red\_color = 4'b0000;  blue\_color = 4'b0000;  green\_color = 4'b0000;  end    end  endmodule |

## Приложение Б1



**Рисунок Б1 – Временная диаграмма теста vga**

## Приложение В1

*Листинг В1 — описание тест-модуля VGA*

|  |
| --- |
| module vga\_test();  reg clk;  reg rst;  reg [2:0] sw;    wire h\_sync, v\_sync;  wire [2:0] rgb ;    initial  begin  rst =0;  clk =0;  sw =3'b010;  end    always # 5 //100 MHz  clk = ~clk;    vga\_top uut (  .clk(clk),  .reset(rst),  .sw(sw),  .hsync(h\_sync),  .vsync(v\_sync),  .rgb(rgb)  );      endmodule |

## Приложение Д1

*Листинг Д1— файл проектных ограничений*

|  |
| --- |
| create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports{clk}]  set\_property IOSTANDARD LVCMOS33 [get\_ports { clk }]  set\_property PACKAGE\_PIN E3 [get\_ports { clk }]  set\_property PACKAGE\_PIN B11 [get\_ports { hsync }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { hsync }]  set\_property PACKAGE\_PIN B12 [get\_ports { vsync }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { vsync }]  set\_property PACKAGE\_PIN A4 [get\_ports { VGA\_R\_3 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_R\_3 }]  set\_property PACKAGE\_PIN C5 [get\_ports { VGA\_R\_2 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_R\_2 }]  set\_property PACKAGE\_PIN B4 [get\_ports { VGA\_R\_1 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_R\_1 }]  set\_property PACKAGE\_PIN A3 [get\_ports { VGA\_R\_0 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_R\_0 }]  set\_property PACKAGE\_PIN A6 [get\_ports { VGA\_G\_3 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_G\_3 }]  set\_property PACKAGE\_PIN B6 [get\_ports { VGA\_G\_2 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_G\_2 }]  set\_property PACKAGE\_PIN A5 [get\_ports { VGA\_G\_1 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_G\_1 }]  set\_property PACKAGE\_PIN C6 [get\_ports { VGA\_G\_0 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_G\_0 }]  set\_property PACKAGE\_PIN D8 [get\_ports { VGA\_B\_3 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_B\_3 }]  set\_property PACKAGE\_PIN D7 [get\_ports { VGA\_B\_2 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_B\_2 }]  set\_property PACKAGE\_PIN C7 [get\_ports { VGA\_B\_1 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_B\_1 }]  set\_property PACKAGE\_PIN B7 [get\_ports { VGA\_B\_0 }]  set\_property IOSTANDARD LVCMOS33 [get\_ports { VGA\_B\_0 }] |

## Приложение Д2

Руководство пользователя:

1.Чтобы конвертировать изображения нужно использовать программу, представленную в Приложении А1.

2.После конвертации, стоит выгрузить значения из текстовых файлов в соответствующие блоки RAM, в Сапр vivado.

3.После генерации, нового bit-файла загрузить на ПЛИС.