Домашнее задание №2 по курсу:

«Введение в архитектуру вычислительных систем»

Вариант 1

В данном задании Вы на практике познакомитесь с тем, что из себя представляет RTL разработка. Компиляция и запуск Verilog-исходников будут производиться с помощью программы с открытым исходным кодом «Icarus Verilog» под лицензией GNU GPL. Просмотр временных диаграмм будет производиться в программе с открытым исходным кодом «GTKWave».

Часть 1. Установка Icarus Verilog и GTKWave

Linux Ubuntu

1. Установите Icarus Verilog
    - В командной строке
      ```
      sudo apt install iverilog
      ```
2. Установите GTKWave
    - В командной строке
      ```
      sudo apt install gtkwave
      ```

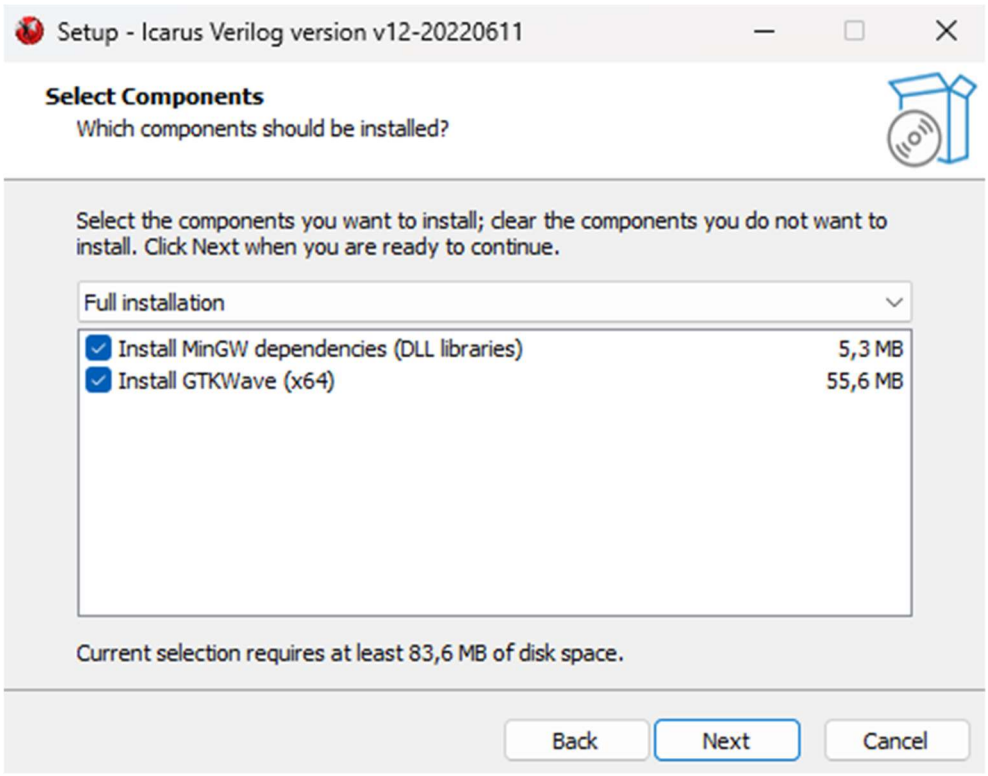
MacOS

1. Установите Icarus Verilog
    - В командной строке
      ```
      brew install icarus-verilog
      ```
2. Установите GTKWave
    - В командной строке
      ```
      brew install gtkwave
      ```

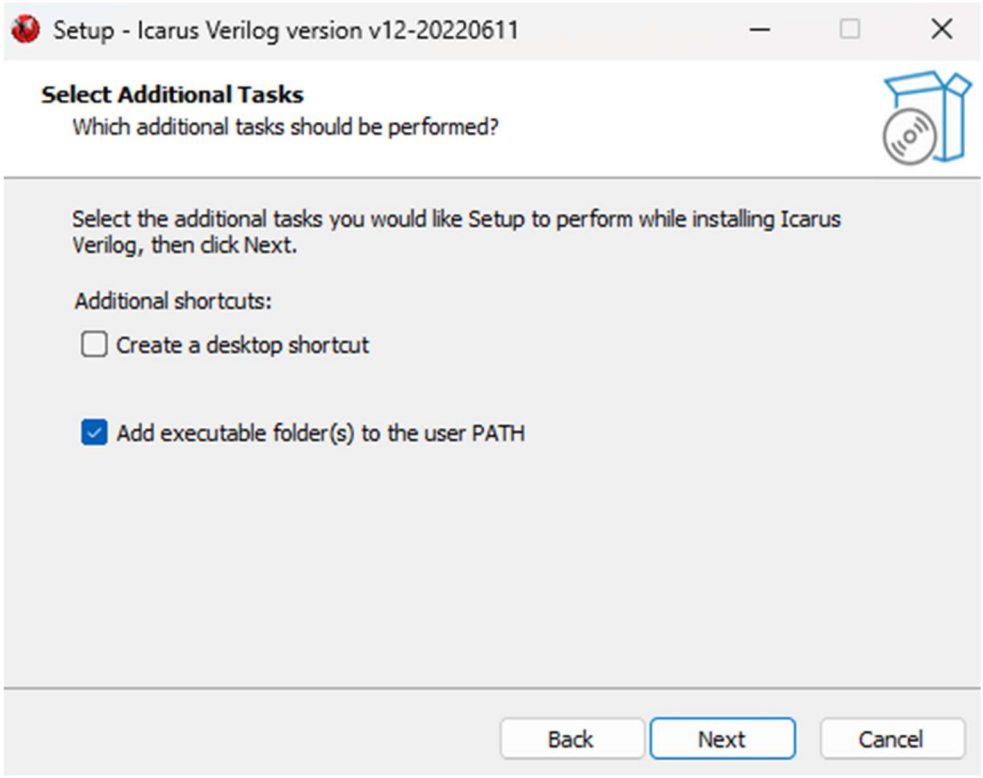
Windows

1. Скачайте Icarus Verilog & GTKWave
    - Файл доступен для скачивания по ссылке
    - Рекомендуемый файл для скачивания «iverilog-v12-20220611-x64_setup [18.2MB]»
    - В данной версии Icarus Verilog и GTKWave включены в один файл для установки

2. Установите Icarus Verilog & GTKWave
    - Запустите скачанный файл для установки
    - Пройдите по шагам программы для установки
        i. Убедитесь, что выставлены обе галочки: «**Install MinGW dependencies (DLL libraries)**» и «**Install GTKWave (x64)**»



        ii. Убедитесь, что выставлена галочка «**Add executable folder(s) to the user PATH**»



Часть 2. Работа с Icarus Verilog и GTKWave

Имплементация RTL кода

- Для имплементации RTL кода может быть использован любой удобный для Вас редактор, например, Visual Studio Code
    - Для подсветки синтаксиса языка Verilog рекомендуется расширение «TerosHDL» (teros-technology.teroshdl), доступное в Visual Studio Code
- Создайте файл с расширением «**.v**» для описания модуля
    - Например, рассмотренный в лекции «**mux4.v**» – мультиплексор четырех входных каналов в один выходной

```
// File mux4.v
module mux4 #(

  parameter WIDTH = 8
)(
  input  [WIDTH-1:0] d0_i,
  input  [WIDTH-1:0] d1_i,
  input  [WIDTH-1:0] d2_i,
  input  [WIDTH-1:0] d3_i,
  input  [1:0]       sel_i,
  output [WIDTH-1:0] res_o
);

  reg [WIDTH-1:0] res_internal;

  always @(*) begin
    res_internal = {WIDTH{1'b0}};
    case (sel_i)
      2'b00: res_internal = d0_i;
      2'b01: res_internal = d1_i;
      2'b10: res_internal = d2_i;
      2'b11: res_internal = d3_i;
    endcase
  end

  assign res_o = res_internal;

endmodule
```

- Создайте файл с расширением «**.v**» для описания testbench
  - Например, testbench «**mux4_tb.v**» для рассмотренного в лекции «**mux4.v**»

```
// File mux4_tb.v
`timescale 1ns/1ps

module mux4_tb();
  // mux4 localparam
  localparam WIDTH = 8;
  // mux4 ports
  // `reg` for inputs
  // `wire` for outputs
  reg  [WIDTH-1:0] d0_i;
  reg  [WIDTH-1:0] d1_i;
  reg  [WIDTH-1:0] d2_i;
  reg  [WIDTH-1:0] d3_i;
  reg  [1:0]       sel_i;
  wire [WIDTH-1:0] res_o;
  // mux4 instance
  mux4 #(
    .WIDTH(WIDTH)
  ) mux4_inst (
    .d0_i (d0_i ),
    .d1_i (d1_i ),
    .d2_i (d2_i ),
    .d3_i (d3_i ),
    .sel_i(sel_i),
    .res_o(res_o)
  );
  // testbench logic
  initial begin
    $dumpvars; // For waveforms
    d0_i = 8'hAA;
    d1_i = 8'hBB;
    d2_i = 8'hCC;
    d3_i = 8'hDD;
    sel_i = 2'b00;
    #10;
    sel_i = 2'b01;
    #10;
    sel_i = 2'b10;
    #10;
    sel_i = 2'b11;
    #10;
    $finish;
  end
endmodule
```

Компиляция и запуск кода

Linux Ubuntu / MacOS

- Запустите терминал в директории с готовыми Verilog файлами
- Для компиляции файлов с помощью Icarus Verilog необходимо выполнить команду
  `iverilog <file1.v> <file2.v> <file3.v> …`
  - Verilog файлы необходимо указывать, в соответствии с иерархией разрабатываемого модуля, в порядке включения модулей друг в друга
  - Например, для приведенного в примере «mux4»
    `iverilog ./mux4.v ./mux4_tb.v`
- В результате компиляции сгенерируется исполняемый файл «**a.out**»
- Для генерации временных диаграмм необходимо запустить сгенерированный исполняемый файл, исполнив команду
  `./a.out`
- В результате успешной генерации временных диаграмм, будет сгенерирован файл «**dump.vcd**» в командной строке будет выведено сообщение
  `VCD info: dumpfile dump.vcd opened for output.`
- Для просмотра временных диаграмм необходимо открыть сгенерированный файл с временными диаграммами в GTKWave, исполнив команду:
  `gtkwave dump.vcd`
- В результате откроется интерфейс программы GTKWave

Windows

- Запустите PowerShell в директории с готовыми Verilog файлами
- Для компиляции файлов с помощью Icarus Verilog необходимо выполнить команду
  `iverilog.exe <file1.v> <file2.v> <file3.v> …`
  - Verilog файлы необходимо указывать, в соответствии с иерархией разрабатываемого модуля, в порядке включения модулей друг в друга
  - Например, для приведенного в примере «mux4»
    `iverilog.exe .\mux4.v .\mux4_tb.v`
- В результате компиляции сгенерируется исполняемый файл «**a.out**»

- Для генерации временных диаграмм необходимо запустить сгенерированный исполняемый файл, исполнив команду
  `vvp.exe .\a.out`
- В результате успешной генерации временных диаграмм, будет сгенерирован файл «**dump.vcd**» в командной строке будет выведено сообщение
  `VCD info: dumpfile dump.vcd opened for output.`
- Для просмотра временных диаграмм необходимо открыть сгенерированный файл с временными диаграммами в GTKWave, исполнив команду:
  `gtkwave.exe .\dump.vcd`
- В результате откроется интерфейс программы GTKWave

Просмотр временных диаграмм в GTKWave

- Интерфейс GTKWave выглядит так, как представлено на рисунке 1 ниже

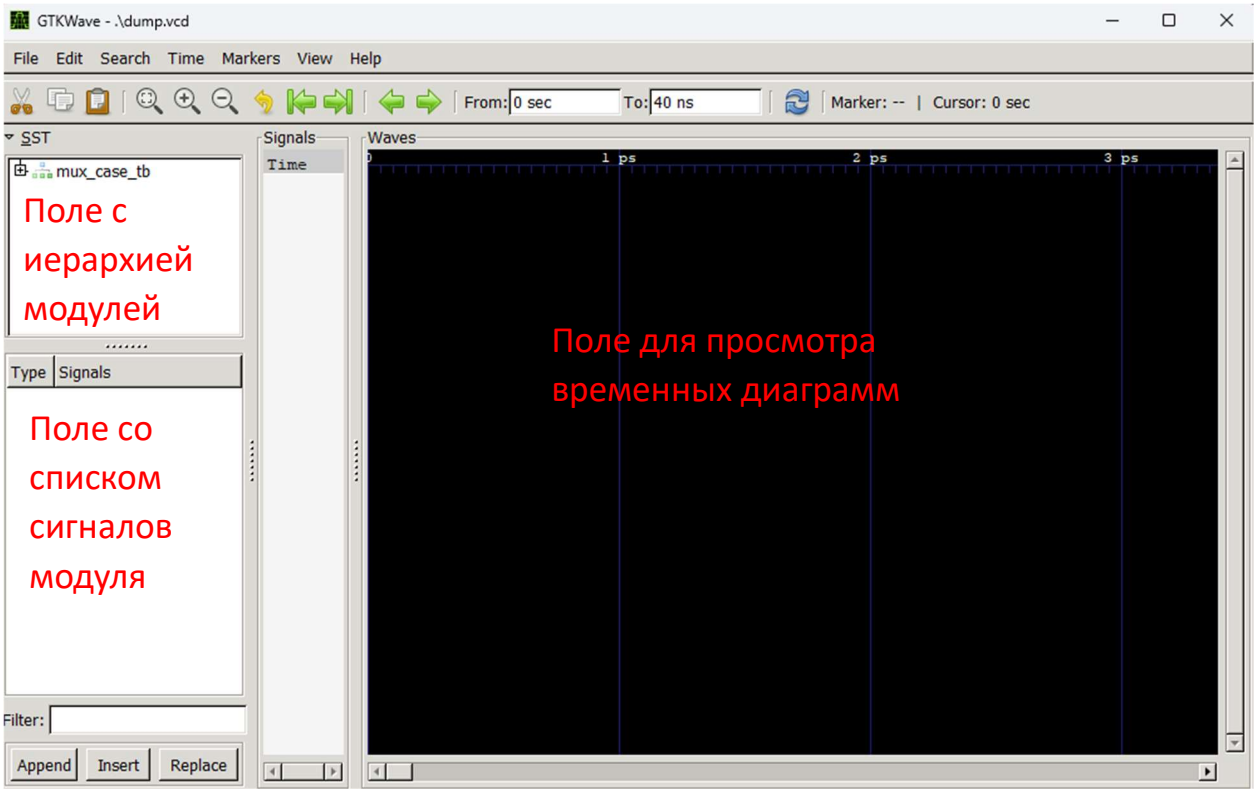


Рисунок 1. Интерфейс при запуске GTKWave

- Раскрыть иерархию модулей, нажав «**+**» в поле с иерархией модулей

- Выбрать желаемый модуль для просмотра временных диаграмм с его шинами и сигналами
    - Например, рассматриваемый в примере «**mux4_inst**», в результате чего в поле со списком сигналов модуля отобразятся все сигналы модуля **mux4** (рисунок 2)

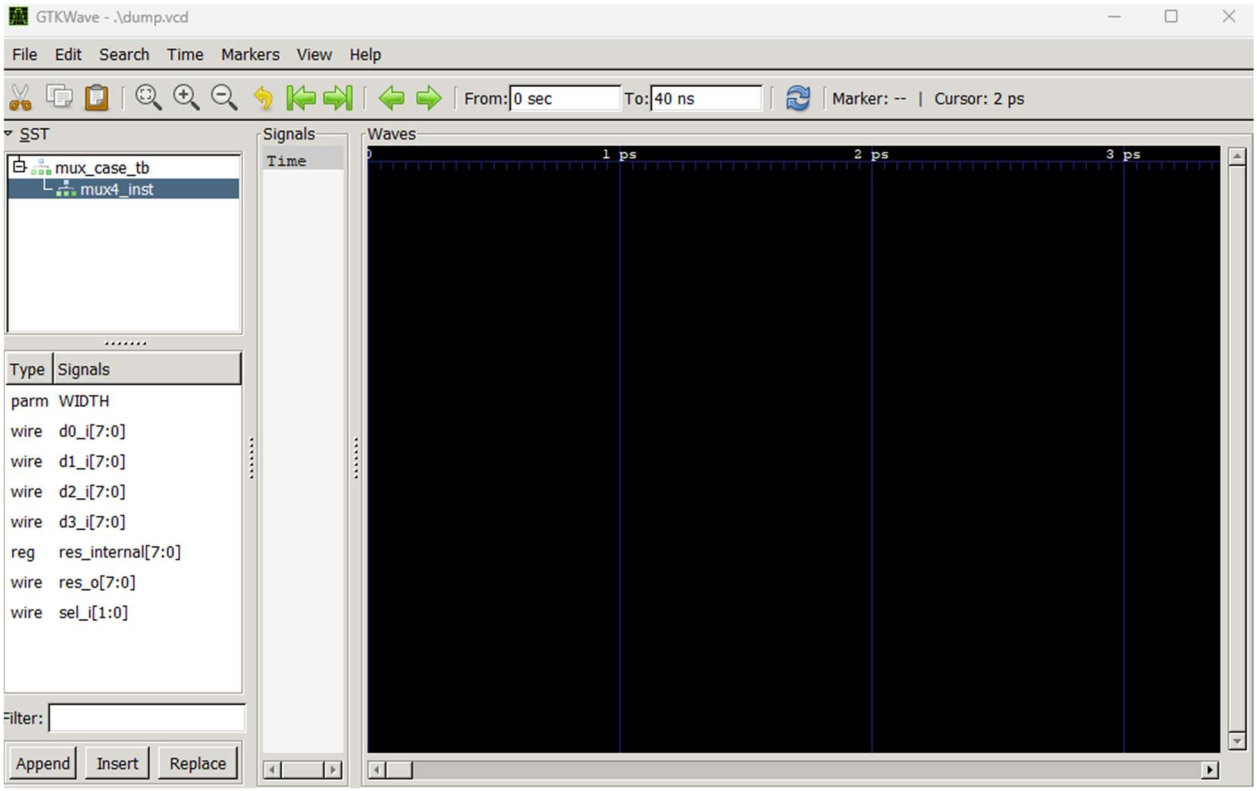


Рисунок 2. Сигналы модуля mux4

- Для просмотра сигналов выбранного модуля, в поле со списком сигналов модуля необходимо выделить желаемые сигналы и нажать кнопку «**Append**» -> в результате временные диаграммы для выбранных сигналов отобразятся в поле для просмотра временных диаграмм (рисунок 3)
    - Для наглядности рекомендуется отмасштабировать отображение временных диаграмм на весь экран, с помощью нажатия кнопки «**Zoom Fit**»

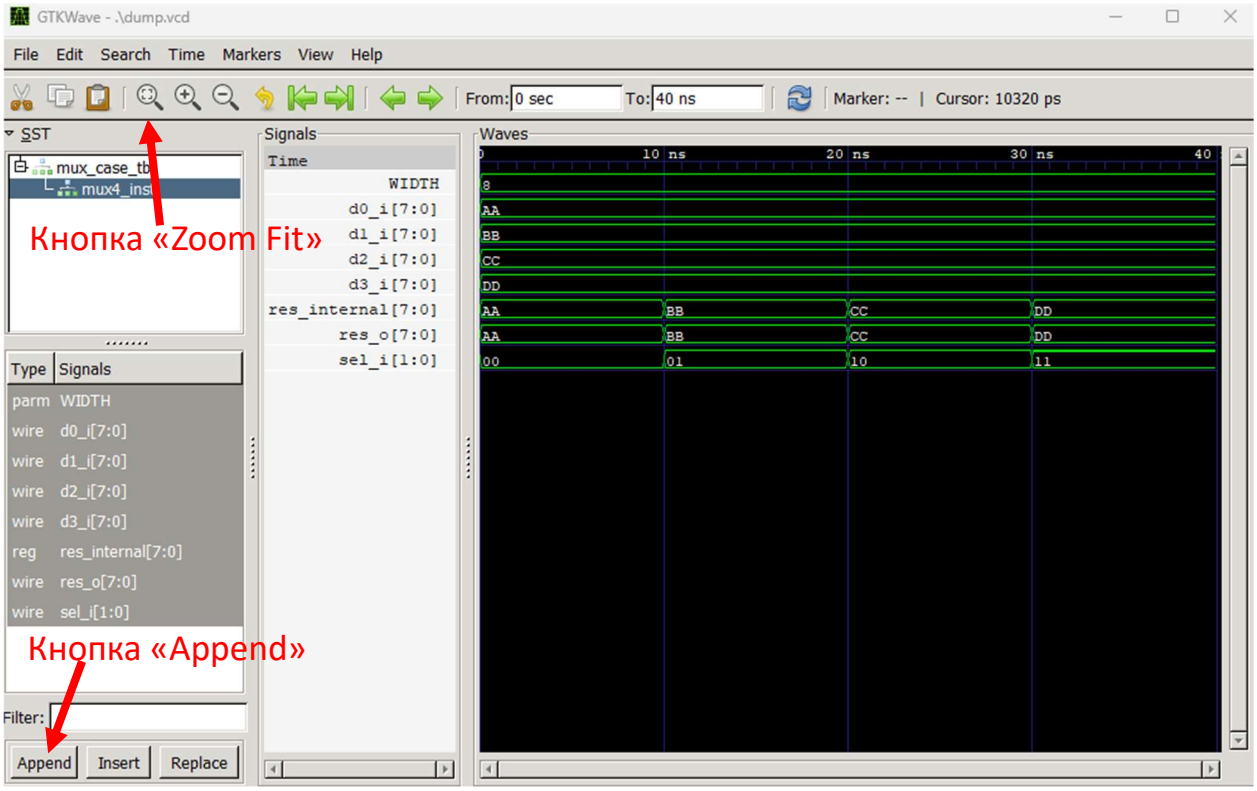


Рисунок 3. Временные диаграммы для модуля mux4

Часть 3. Задание

Вам необходимо разработать, имплементировать и протестировать модуль **alu_register**, выполняющий различные арифметические операции и записывающий результат в регистр, в соответствии с требованиями

- Модуль имеет параметр **WIDTH** для указания ширины операндов
- Модуль имеет следующие порты

Название	Ширина	Направление	Описание
clk_i	1	Input	Тактовый сигнал
rst_i	1	Input	Сигнал сброса
first_i	WIDTH	Input	Шина первого операнда
second_i	WIDTH	Input	Шина второго операнда
opcode_i	3	Input	Шина, кодирующая операцию
result_o	WIDTH	Output	Шина результата операции

- Модуль включает регистр (D-триггер) для сохранения результата операции: Результат операции, исполняемой в текущий такт, сохранен в регистре на следующий такт
    - Шина **result_o** отображает значение, записанное в регистре
    - Регистр сбрасываемый, значение при сбросе: 0
- Модуль тактируется сигналом **clk_i**, тактирование по положительному фронту
- Сброс синхронный, по активному высокому уровню сигнала **rst_i**
- Арифметические операции, в зависимости от значения **opcode_i**

opcode_i	Операция над first_i и second_i
3’b000	Побитовое НЕ-И
3’b001	Побитовое Исключающее ИЛИ
3’b010	Сложение (без знака)
3’b011	Арифметический (знаковый) сдвиг first_i вправо на second_i
3’b100	Побитовое ИЛИ
3’b101	Логический сдвиг first_i влево на second_i
3’b110	Побитовое НЕ для first_i
3’b111	first_i < second_i

Пункт 1. Документация

Подготовьте документацию для разрабатываемого модуля, включающую следующие разделы:

- **Функциональное описание**, в котором необходимо кратко описать функционал разрабатываемого модуля, в соответствии с требованиями
- **Структурная схема**, в котором необходимо представить структурную схему разрабатываемого модуля
    - Для рисования структурной схемы рекомендуется использовать свободное ПО draw.io
- **Параметры**, в котором необходимо привести описание всех параметров разрабатываемого модуля и их допустимые значения
- **Порты**, в котором необходимо привести описание всех портов разрабатываемого модуля и их ширину
- **Тактирование и сброс**, в котором привести описание, того, какими сигналами осуществляется тактирование и сброс разрабатываемого модуля, а также описать процедуру сброса
- **Тестирование**, в котором привести описание сценария, который будет реализован при тестировании разрабатываемого модуля

Пункт 2. Имплементация

Имплементируйте разрабатываемый модуль на языке Verilog, в соответствии с требованиями

- Вы можете как имплементировать всю логику разрабатываемого модуля в одном Verilog-модуле, так и разделить логику на несколько Verilog-модулей, а затем инстанцировать отдельные логические компоненты в итоговом Verilog-модуле

Пункт 3. Тестирование

Имплементируйте testbench на языке Verilog для тестирования и отладки разрабатываемого модуля

- Testbench должен включать процедуру сброса разрабатываемого модуля
- Testbench должен иллюстрировать весь функционал разрабатываемого модуля, в соответствии с требованиями

Используйте временные диаграммы для отладки разрабатываемого модуля

Часть 4. Отчет

Отчет должен быть оформлен в виде Word/PDF файла и отправлен на почту khajdukov.di@phystech.edu ответным письмом.

ДЕДЛАЙН 31.03.2025 в 23:59 МСК (через 2 недели).

Структура отчета:

1. ФИО
2. Номер группы
3. Номер варианта
4. Email почты phystech.edu
5. Ответы, в соответствии с заданием
    a. Документация для разработанного модуля
    b. Файл (или файлы) с расширением «**.v**» с имплементацией разработанного модуля
    c. Файл с расширением «**.v**» с testbench для разработанного модуля
    d. Скриншот (или скриншоты) временных диаграмм, полученные с помощью разработанного testbench

При подозрении на списывание **обе работы будут аннулированы**.

Дополнительные материалы

1. ***Курс от студентов ФРКТ по языку Verilog на YouTube***
   https://www.youtube.com/playlist?list=PLhtMaaf_npBz9zfsJMZC12Lk3zvHiJckr
2. Харрис Д. М., Харрис Д. Цифровая схемотехника и архитектура компьютера RISC-V. – 2022.
   4 глава книги посвящена языкам описания аппаратуры, в частности SystemVerilog
3. Томас Д. Логическое проектирование и верификация систем на SystemVerilog. – 2019.
   Книга, полностью посвященная языку SystemVerilog

Успехов!