Руководство по выполнению лабораторной работы day9_lab2

1 Откройте Visual Studio Code. Перейдите в каталог **ce2020labs/day_9/lab2**

2 Проведите подготовку каталогов проекта

2.1 Проведите обзор каталогов проекта.

2.1.1 Каталог **lab2** содержит следующие каталоги

- **common** — общие компоненты для проекта ПЛИС
- **program** — каталог для сборки программ процессора schoolRISK-V
- **run_rzrd** — каталог для сборки проекта ПЛИС
- **school_risk** — каталог с компонентами процессора schoolRISC-V
- **src_calc** — каталог с рабочими компонентами, если там есть файл **example_cpu.sv** то удалите его
- **src_rzd** — каталог с файлами для верхнего уровня проекта ПЛИС
- **src_tb** — каталог с файлами для симуляции, если там есть файл **tb.s**v то удалите его
- **support —** каталог с рабочими файлами для разных шагов лабораторной работы
- может быть каталог **work**, если он есть то его следует удалить

2.1.2 Каталог **program** содержит следующие каталоги и файлы

- **common** — общие файлы для сборки программ
- **p0_program** — каталог для сборки программы процессора P0

  Если внутри каталога **p0_program** находится файл **main.S** то его следует удалить

2.1.3 Каталог **support** содержит следующие каталоги

- **full** — рабочие файлы для полного варианта лабораторной работы
- **step1** — рабочие файлы для шага 1

2.1.4 Каталог **lab2** содержит следующие файлы:

- **systemverilog.txt** — файл со списком файлов для моделирования
- **vlib_init.sh** — инициализация системы моделирования
- **compile.sh** — компиляция файлов для моделирования
- **c_run_0.sh** — запуск моделирования в консольном режиме
- **g_run_0.sh** — запуск моделирования в режиме GUI
- **run_all.sh** — запуск компиляции и выполнения нескольких тестов в консольном режиме

2.2 Откройте терминал в программе Visual Studio Code

2.3 Выполните скрипт **./vlib_init.sh**

Будет создан каталог **work** с пустой рабочей библиотекой.

2.4 Выполните скрипт ./compile.sh

Вы должны получить ошибку, в проекте отсутствуют файлы **src_calc/example_cpu.sv** и **src_tb/tb/sv**

**** Error: (vlog-7) Failed to open design unit file "src_calc/example_cpu.sv" in read mode.**

**** Error: (vlog-7) Failed to open design unit file "src_tb/tb.sv" in read mode.**

3 Выполните шаг 1 лабораторной работы

3.1 Скопируйте файл **support/step1/example_cpu.sv** в каталог **src_cal**

cp support/step1/example_cpu.sv src_calc/

3.2 Скопируйте файл **support/step1/tb.sv** в каталог **src_tb**

cp support/step1/tb.sv src_tb/

3.3 Скопируйте файл **support/step1/p0_program/main.S** в каталог **program/p0_progra**m

cp support/step1/p0_program/main.S program/p0_program/

3.4 Проведите обзор файла example_cpu.sv

3.4.1 Найдите следующие компоненты: процессор P0, память для процессора P0, видеопамять, узел vga, память текстур.

3.4.2 Определите как формируются разряды **display_number[15:0]**, какие регистры должны быть записаны.

3.5 Определите место где производится назначение сигнала **index**. Обратите внимание, что в зависимости от текущих координат производится назначение кода 0x32 для извлечения текстуры из видеопамяти.

3.6 Определите место где формируется сигнал rgb для вывода на дисплей.

3.7 Обратите внимание, что на этом этапе процессор не участвует в выводе информации на дисплей.

3.8 Проведите сборку проекта ПЛИС и загрузите его на плату с использованием скрипта ./x_

3.9 Проведите обзор файла **p0_program/main.S**

3.9.1 Определите место где происходит запись в видеопамять

3.9.2 Определите место где выводится цифра на дисплей

3.10 Проведите обзор файлы src_tb/tb.sv

3.10.1 Определите условие по которому производится решение об успешном выполнении теста. Какой выбран сигнал для контроля теста и какое значение он должен иметь ?

3.10.2 Выполните сборку программы с помощью команды **make board** в каталоге **program/p0_program**

3.11 Проведите компиляцию проекта **./x_synthesize.bash** в каталоге **run_rzrd**

3.12 На экране должна отображаться цифра 2 примерно в центре экрана.

3.13 В строке где присваивается значение сигналу **index** измените код символа и координаты. Проведите повторную сборку прокекта. Что изменилось на экране ?

4 Выполните шаг 2 лабораторной работы

4.1.1 В файле **example_cpu.sv** закомментируйте строчку с назначением сигнала **index**

4.1.2 Уберите комментарий в фрагменте кода между строчками Step 2 и Step 2 — end. В этом фрагменте кода подключается видеопамять с отключённым портом записи.

4.1.3 Соберите проект. Экран должен быть заполнен буквой A

4.1.4 Обратите внимание, компонент **video_memory** производит начальное заполнение памяти выбранным значением. Измените начальное значение на 0x20, это символ пробела. Проведите сборку проекта. Что изменилось на экране ?

4.2 Выполните шаг 3 лабораторной работы

4.2.1 В файле **example_cpu.sv** закомментируйте строчки с назначением нулевых значений для сигналов **video_w_addr, video_w_data, video_w_valid**

4.2.2 Уберите комментарий в фрагменте кода между строчками Step 3 и Step 3 — end. В этом фрагменте кода подключается процессор к порту записи в видеопамять.

4.2.3 Проведите сборку проекта. На экране должна быть строчка «Hello»

4.3 Выполните шаг 4 лабораторной работы

4.3.1 Откройте файл **program/p0_program/main.S**

4.3.2 Определите место где должна производится запись слова «World!» и уберите комментарий с команд записи.

4.3.3 Проведите сборку программы и проекта ПЛИС. На экране должна отображаться строчка «Hello World!»

4.4 Выполните шаг 4 лабораторной работы

4.4.1 Запустите скрипт **./complile.sh** компиляция должна пройти без ошибок. Обратите внимание, что при компиляции выводится много сообщений.

4.4.2 Повторно запустите компиляцию командой: **./compile.sh | grep Error**

Компиляция должна пройти без ошибок. Строчка «Error» должна быть выделена красным цветом. Обратите внимание, что в данном случае выводится только одна строка. Если бы в исходных файлах были ошибки, то они тоже были бы выведены.

4.4.3 Запустите моделирование в консольном режиме командой **./c_run_0.sh**

Тест должен завершиться с ошибкой:

test_id=0 test_name: test_0 DEPTH=8 TRANSACTION=14 TEST_FAILED *****

4.4.4 Запустите симулятор в режиме GUI командой **./g_run_0.sh &**

Обратите внимание на символ & после команды. Этот символ даёт указание запустить программу и вернуть управление терминалу. В терминале можно вводить другие команды, в частности можно запускать скрип компиляции ./compile.sh

4.4.5 В симуляторе выведите на временную диаграмму сигналы верхнего уровня:

- **reset_p**
- **display_number**

4.4.6 Во второе окно с временной диаграммой выведите сигналы компонента **uut**

4.4.7 В третье окно с временной диаграммой выведите сигналы компонента **video_memory**

4.4.8 Запустите сеанс моделирования на время 90 us. В окне «Transcript» будет выведен лог теста, там также должно быть выведено сообщение «TEST FAILED»

4.4.9 Какое значение в конце моделирования имеет сигнал выбранный для контроля правильности проведения теста ? (см. п. 3.10.1)

4.4.10 Что нужно сделать для правильного завершения теста ?

4.4.11 Измените файл **tb.sv** для правильного завершения теста

4.4.12 Скомпилируйте заново файлы проекта, для этого перейдите в терминал #1 и выполните скрипт **./compile.sh**

Обратите внимание, что не требуется закрывать программу симулятора. Скрипт **./compile** также можно выполнить в окне «Transcrip» симулятора. Однако у нас основной системой разработки является Visual Studio Code. В случае появления ошибок при компиляции есть возможность сразу перейти на строку с ошибкой и её исправить. Это удобно.

4.4.13 Перейдите в симулятор. Выполните перезапуск сеанса. Запустите сеанс моделирования на время 90 us

4.4.14 Какой результат выполнения теста ? Если тест завершился с ошибкой, то вернитесь к п. 4.4.11

4.4.15 Запустите моделирование в консольном режиме командой **./c_run_0.sh**

Тест должен завершиться успешно:

 # test_id=0 test_name: test_0 DISPLAY_NUMBER=0xabcd TEST_PASSED

4.4.16 Обратите внимание, что в проекте реализован тест с самопроверкой. Результатом теста является сообщение «TEST PASSED» или «TEST FAILED».

Тест может быть запущен в консольном режиме или в режиме GUI. Режим GUI позволяет производить отладку проекта, в том числе пошаговую отладку. Консольный режим позволяет провести быстрый запуск теста и даёт возможность для удобного запуска группы тестов.

4.5 Дополнительный шаг лабораторной работы

4.5.1 Обратите внимание, на экране должна быть строчка «Hello,World!». Должна быть запятая между словами. В компоненте **text_rom** нет текстуры для символа запятой. Дополните **text_rom** текстурой для символа запятой.

Благодарю за выполнение лабораторной работы!

Буду рад получить отзывы, замечания, предложения по данной работе.

С уважением,

Дмитрий Смехов

dsmekhov@plis2.ru