Руководство по выполнению лабораторной работы day6_lab1

1 Откройте Visual Studio Code. Перейдите в каталог **ce2020labs/day_6/lab1**

2 Проведите подготовку каталогов проекта

2.1 Проведите обзор каталогов проекта.

2.1.1 Каталог **lab1** 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_risc** — 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
- **p1_program** — каталог для сборки программы процессора P0

  Если внутри каталогов **p0_program**, **p1_program** находятся файлы **main.S** то их 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 **lab1** 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
- **p_build.sh** — сборка программ для процессоров P0 и P1

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 отсутствует файл **src_calc/example_cpu.sv**

**** Error: (vlog-7) Failed to open design unit file "src_calc/example_cpu.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 Скопируйте файл **support/step1/p1_program/main.S** в каталог **program/p1_program**

cp support/step1/p1_program/main.S program/p1_program/

3.5 Проведите обзор файла example_cpu.sv

3.5.1 Найдите следующие компоненты: процессор P0, процессор P1, память для процессора P0, память для процессора P1

3.5.2 Определите как формируются разряды **display_number[11:8]**, какой процессор из формирует и какие регистры должны быть записаны.

3.5.3 Определите как формируются разряды **display_number[3:0],** какой процессор из формирует и какие регистры должны быть записаны.

3.5.4 Определите в какой регистр попадает информация о нажатии клавиши **key_sw_p[3]**. В какой процессор попадает информация о нажатии клавиши, какие регистры участвуют в опросе клавиши.

3.5.5 Определите в какой регистр попадает информация о нажатии клавиши **key_sw_p[2]**. В какой процессор попадает информация о нажатии клавиши, какие регистры участвуют в опросе клавиши.

3.5.6 Определите что происходит при нажатии на клавищу **key_sw_p[0]**

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

3.6.1 Определите место где происходит опрос клавиш KEY0, KEY1

3.6.2 Определите место где производится цикл записи в память

3.6.3 Определите место где производится цикл чтения из память

3.6.4 Определите место где выводится цифра результата на дисплей

3.7 Проведите обзор файлы **src_tb/tb.sv**

3.7.1 Определите место где формируются сигналы key_sw_p[2] и key_sw_p[3]; Сколько импульсов формируется для каждого из сигналов ?

3.8 Проведите компиляцию проекта

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

3.8.2 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.

3.9 Выполните сборку программ

3.9.1 Выполните команду **./p_build.sh**

Будет выполнена компиляция программ для процессоров P0, P1 и файлы программы будет скопированы в два каталога:

- ./ - это текущий каталог проекта, там он будет использоваться для моделирования
- ./run_rzrd — это каталог для сборки проекта ПЛИС

3.10 Выполнение моделирования проекта

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

Тест должен завершиться с ошибкой:

test_id=0 test_name: test_0 DISPLAY_NUMBER=0x5ad4 TEST_FAILED *****

3.10.2 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

3.10.3 В симуляторе выведите на временную диаграмму сигналы верхнего уровня:

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

- **test_passed**

3.10.4 В другое окно с временной диаграммой выведите сигналы компонента **uut**

3.10.5 Запустите сеанс моделирования на время 30 us. В окне «Transcript» будет выведен лог теста, там также должно быть выведено сообщение «TEST FAILED»

3.10.6 Какое значение в конце моделирования имеет сигнал выбранный для контроля правильности проведения теста ?

3.10.7 Что нужно сделать для правильного завершения теста ?

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

3.10.9 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.

3.10.10 Перейдите в симулятор. Выполните перезапуск сеанса. Запустите сеанс моделирования на время 30 us

3.10.11 Какой результат выполнения теста ? Если тест завершился с ошибкой, то вернитесь к п. 3.10.8

3.10.12 Перейдите в терминал #1

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

Тест должен завершиться успешно:

test_id=0 test_name: test_0 DEPTH=8 DISPLAY_NUMBER=0x5ad4 TEST_PASSED

3.10.14 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.

3.11 Сборка проекта и загрузка на плату

3.11.1 Подключите плату RZ-EasyFPGA

3.11.2 Откройте новый терминал в Visual Studio Code, это будет терминал #2

3.11.3 Перейдите в каталог **./run_rzrd**

3.11.4 Выполните скрипт **./x_synthesize.bash**

Проект должен быть собран и загружен на плату. На дисплее должны отображаться цифры «0000» и через некоторое время - «0F0F». После этого система готова к обработке нажатия на клавиши.

3.12 Проверка работы на плате

3.12.1 Нажмите на кнопку S1 (крайняя слева). Будет запущен цикл записи на процессоре P0. Крайняя слева цифра должна медленно увеличиваться и дойти до значения 5. Медленное изменение происходит из за очень низкой тактовой частоты, примерно 0.7 Гц. Нажатие на клавишу должно быть достаточно длительным. Должен загореться светодиод 1, он показывает требование записи в память со стороны процессора P0.

3.12.2 Нажмите на кнопку S2 (вторая слева). Будет запущен цикл чтения на процессоре P0. Вторая слува цифра должна стать равной 0xA. Должен мигать светодиод 2, он показывает наличие данных прочитанных из памяти процессором P0.

3.12.3 Нажмите на кнопку S3 (вторая справа). Будет запущен цикл записи на процессоре P1. Вторая справа цифра должна медленно увеличиваться и дойти до значения 0xd. Должен загореться светодиод 3, он показывает требование записи в память со стороны процессора P1.

3.12.4 Нажмите на кнопку S4 (первая справа). Будет запущен цикл чтения на процессоре P1. Первая справа цифра должна стать равной 4. Должен мигать светодиод 3, он показывает наличие данных прочитанных из памяти процессором P1.

3.12.5 Попробуйте одновременно нажать на S2 и S4 что бы одновременно запустить циклы чтения на процессорах P0,P1. Как себя ведут светодиоды ? Почему возможно одновременное свечение светодиодов 1 и 3 ?

3.12.6 Попробуйте одновременно нажать на S1 и S3 что бы одновременно запустить циклы записи на процессорах P0,P1. Как себя ведут светодиоды ? Почему не наблюдается одновременного свечения светодиодов 2 и 4 ?

3.12.7 Попробуйте нажимать на кнопки S1, S2, S3, S4 в произвольном порядке. Проведите анализ изменения цифр.

3.12.8 Влияют ли друг на друга процессоры P0 и P1 которые занимаются опросом кнопок и отображением цифр ?

4 Завершение работы

4.1 Сравните файлы **src_tb/tb.sv, src_calc/example_cpu** с файлами из каталога **support/full**. Какие есть отличия ?

Благодарю за выполнение лабораторной работы!

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

С уважением,

Дмитрий Смехов

dsmekhov@plis2.ru