**Проектирование систем на кристалле с программируемой архитектурой**

**Лабораторная работа №3**

Цели работы:

1. Изучить принципы работы СнК с процессорной ячейкой на базе RISC-V;
2. Имплементировать модуль, разработанный в предыдущих Л/Р в модуль верхнего уровня представленной СнК;
3. Написать программу на языке программирования С, реализующую обращение из процессорного модуля в имплементированный APB3-Slave модуль;
4. Скомпилировать программу и запустить симуляцию проекта в САПР Synopsys VCS, при необходимости добиться корректной работоспособности APB3-Slave модуля.

**Теоретическая часть**

Данная лабораторная работа нацелена на изучение принципа работы общей симуляции большого проекта СнК с интегрированным процессором на базе RISC-V, а также общий принцип проведения «in-system» тестов проекта. Структура рассматриваемого проекта представлена на рисунке 1.

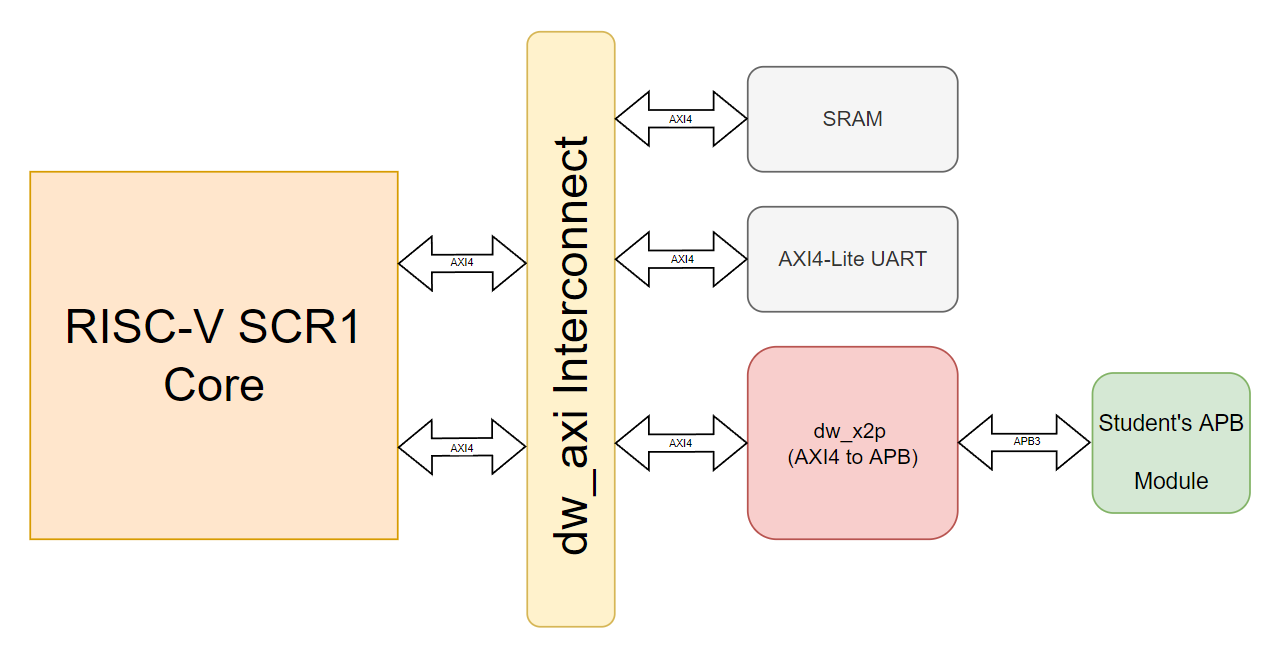


Рисунок 1 – Структура учебного СнК проекта для Л/Р 3

Конструкции языка программирования С, с которыми необходимо ознакомиться:

**Стандартная библиотека stdint.h** широкоиспользуется при написании Embedded ПО для СнК/ПО/FPGA. Позволяет в явном виде задавать переменные различного размера (в битах). К примеру, список доступных переменных из данной библиотеки, которые мы будем использовать в процессе выполнения данной Л/Р.

* *uint8\_t* 8-битовый без знака
* *uint16\_t* 16-битовый без знака
* *uint32\_t* 32-битовый без знака
* *uint64\_t* 64-битовый без знака

**Volatile** — ключевое слово языков C/C++, которое информирует компилятором, что значение переменной может меняться из вне и что компилятор не будет оптимизировать эту переменную. Объекты, объявленные как volatile, не используются в определенных оптимизациях, так как их значения могут изменяться в любое время. При запросе объекта с ключевым словом volatile система всегда считывает его текущее значение, даже если оно запрашивалось в предшествовавшей инструкции. Кроме того, значение объекта записывается непосредственно при присваивании.



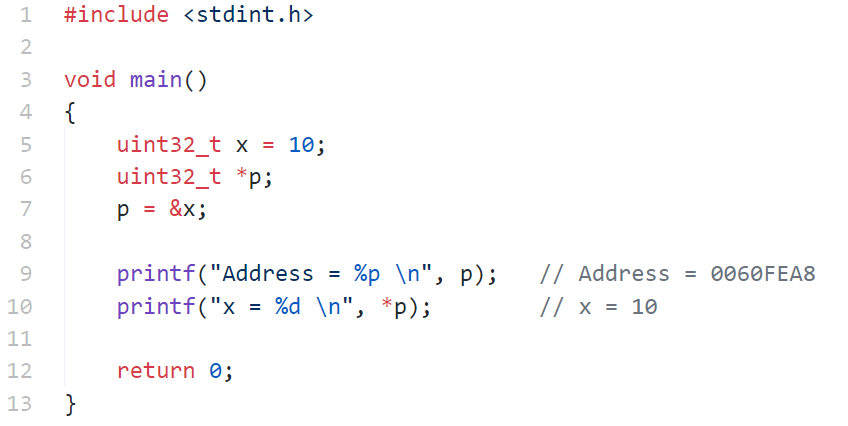
Здесь **statusPtr** указывает на участок памяти, который в любой момент может быть перезаписан. Наша программа, в которой объявлен и проинициализирован этот указатель, не знает, когда это может произойти. От нее тут ничего не зависит. Но благодаря ключевому слову *volatile* можно быть уверенным, что при каждом обращении по этому адресу мы будем получать актуальное изменяемое значение.

**Указатели** представляют собой объекты, значением которых служат адреса других объектов (переменных, констант, указателей) или функций. Указатели - это неотъемлемый компонент для управления памятью в языке Си.

Для работы с указателями в Си определены две операции:

* операция \* (звездочка) — позволяет получить значение объекта по его адресу — определяет значение переменной, которое содержится по адресу, содержащемуся в указателе;
* операция & (амперсанд) — позволяет определить адрес переменной.

Пример использования:



**Структуры** — это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации. Объявление структуры приводит к образованию шаблона, используемого для создания объектов структуры. Переменные, образующие структуру, называются членами структуры. Члены структуры также часто называются элементами или полями. Пример описания структуры для устройства из Л/Р 2 и закреплении за ним определённого адреса памяти представлен на рисунке 2.

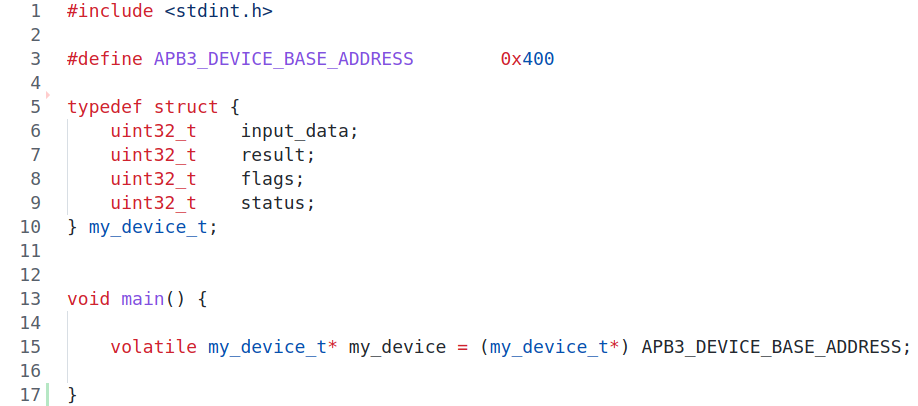


Рисунок 2 – Пример описания структуры и присваивание переменной этой структуры с привязкой к адресу памяти.

**Выполнение лабораторной работы**

1. Необходимо подключить компилятор riscv-unknown-elf-gcc в переменную окружающей среды. Для этого необходимо в файл .bashrc (командой gedit ~/.bashrc) добавить следующую строчку:

export PATH=/local/pkims06/SOC\_PROGRAMMING\_ARCH/riscv-gcc-10.2.0/bin:/$PATH

1. Скопировать директорию, содержащую проект СнК к себе в локальную директорию. Директория располагается по следующему пути:

/local/pkims06/SOC\_PROGRAMMING\_ARCH/soc\_programming

1. Скопировать разработанный в предыдущих Л/Р модуль и интерфейс в локальную директорию из пункта 2 в папку soc\_programming/src/rtl/
2. Интегрировать ваш модуль в файл rtl/top.sv в обозначение комментарием место (Рис. 1). Также необходимо добавить ваш интерфейс APB3 (на примере APB3\_sec) и связать его уже с имеющимся исходным APB3-интерфейсом.

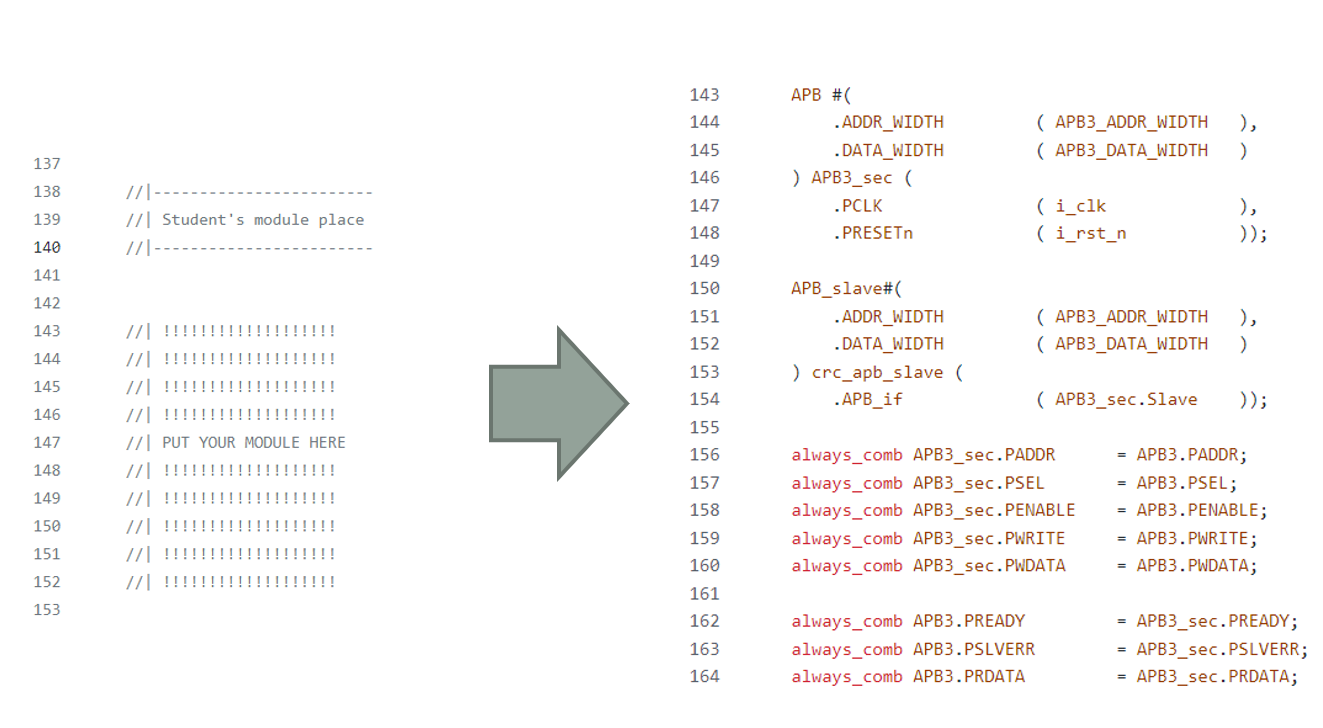


Рисунок 3 – Место инстанцирования модуля и пример его инстанцирования

1. Добавить в файл rtl/rtl\_files.f относительные (***не абсолютные***) пути к файлам из предыдущих ЛР (Рис. 4)

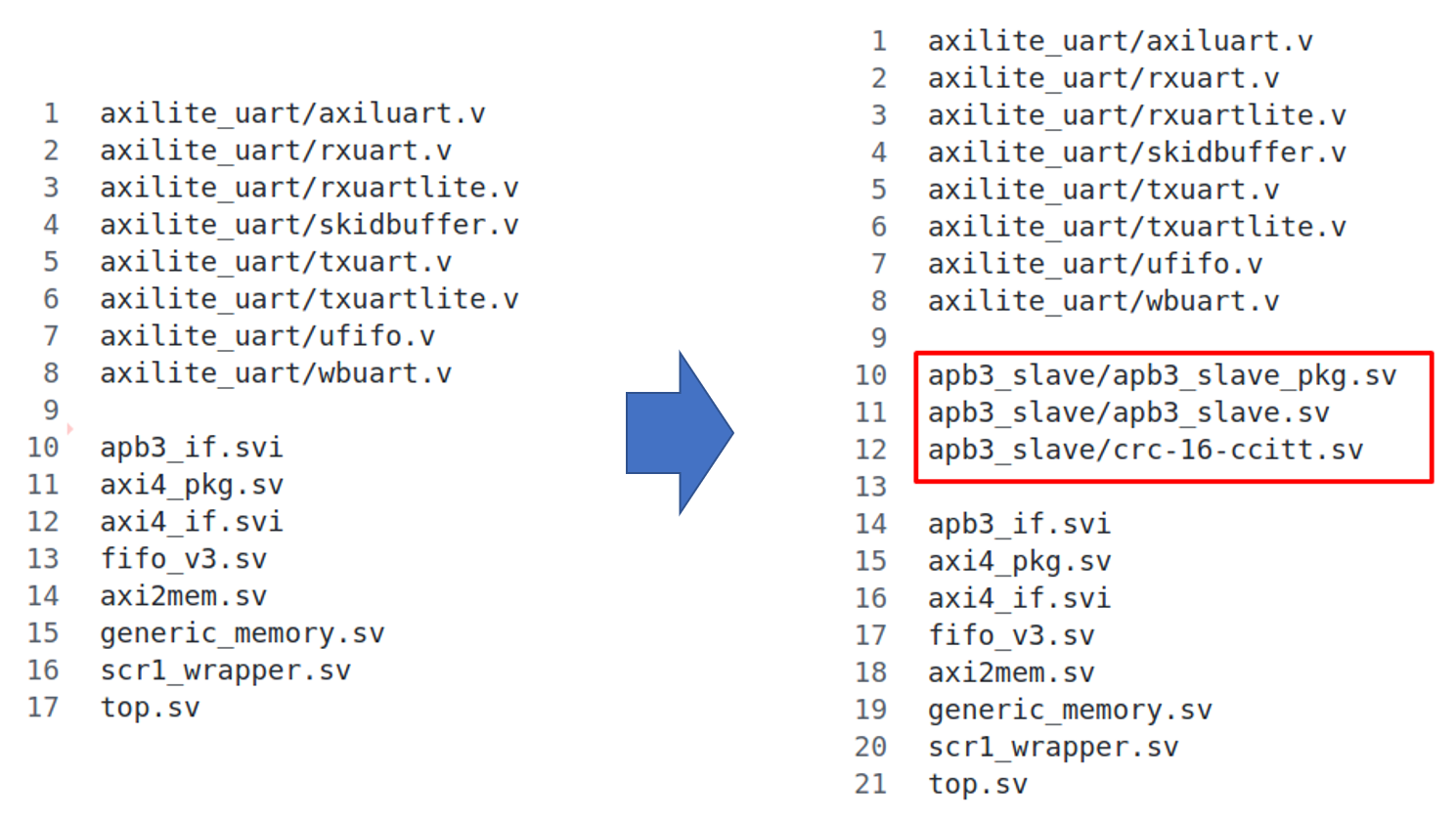


Рисунок 4 – Добавление относительных путей к вашим файлам

1. Написать исполняемый код на ЯП С в файле compile/user\_programm/main.c
2. Скомпилировать код с помощью следующей команды (выполнять из папки compile)

make clean compile

В консоли не должно быть каких-либо ошибок компиляции. Пример успешной компиляции представлен на рисунке 5.

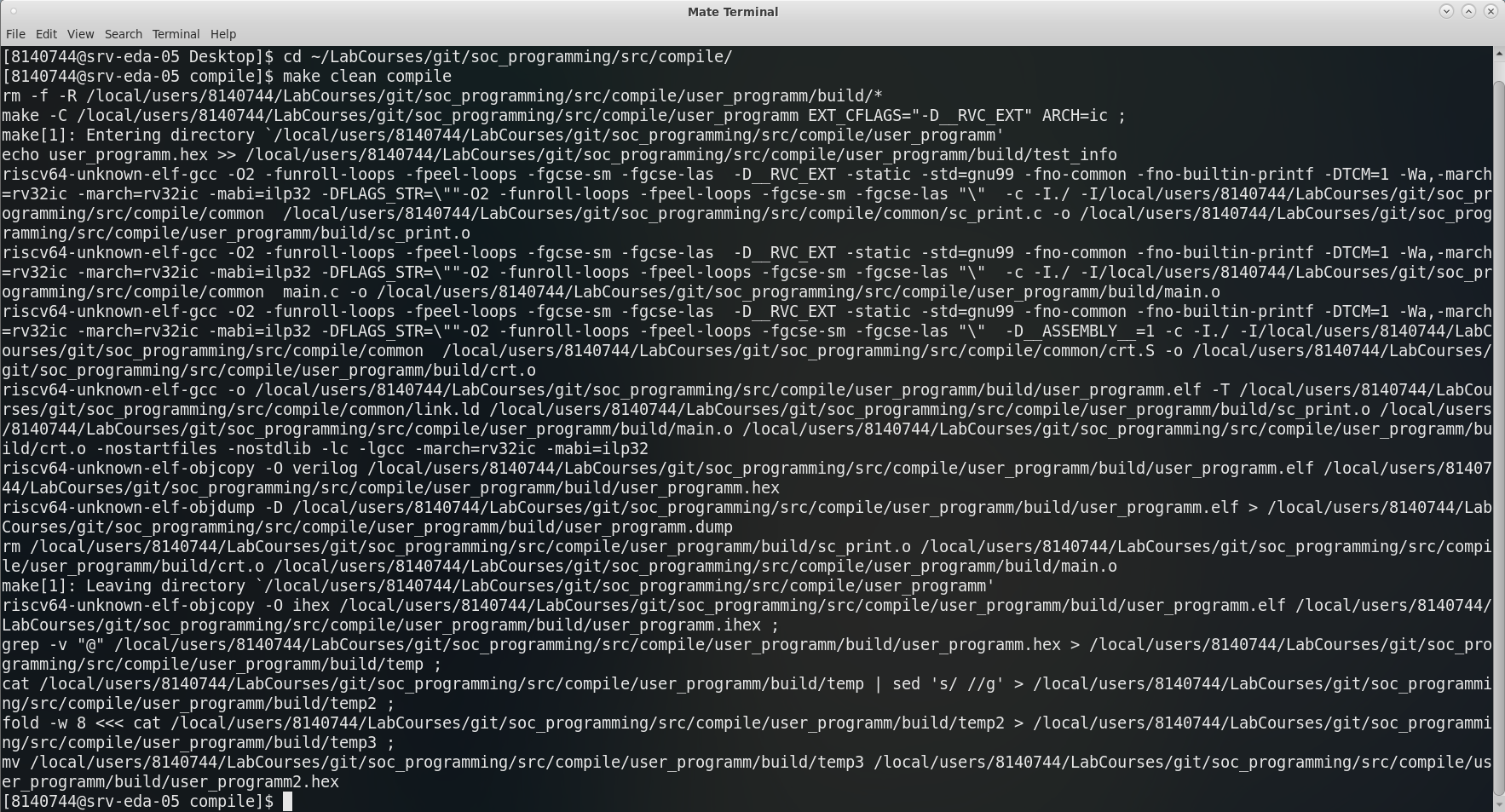


Рисунок 5 – Пример лога компиляции программы без ошибок

1. Подключить модуль Synopsys VCS следующей командой:

module load synopsys/VCS/R-2020.12

1. Перейти в папку simulation и запустить симуляцию проекта в САПР Synopsys VCS с помощью следующей команды:

make clean sim

1. В симуляторе вывести waveform симуляции вашего модуля и продемонстрировать его работоспособность (скриншотами).

**Используемая литература**

1. Программирование на С и С++. Режим доступа: http://www.c-cpp.ru/
2. Syntacore SCR1 RISC-V Core. Режим доступа: <https://github.com/syntacore/scr1>