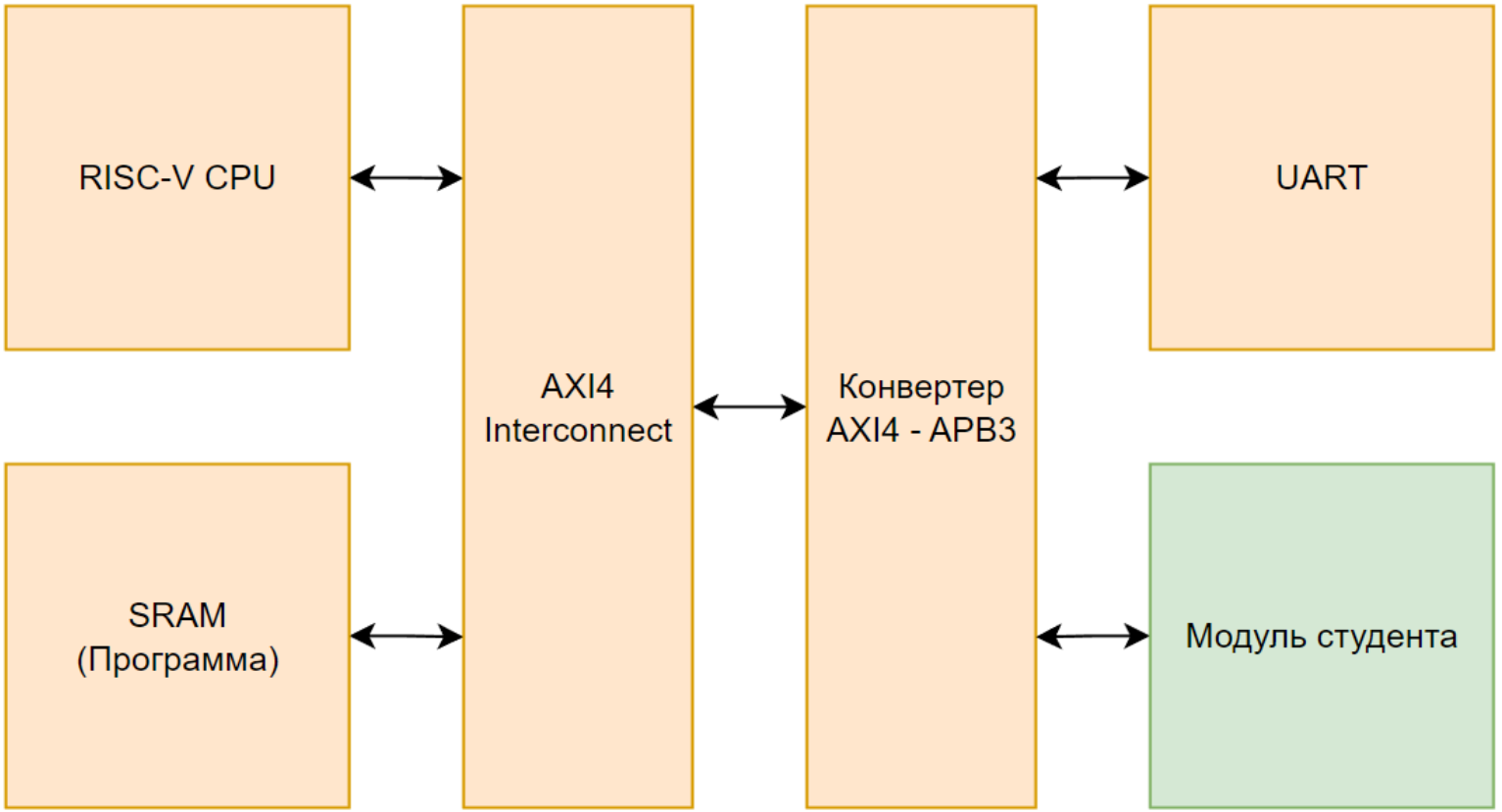
СИСТЕМЫ НА КРИСТАЛЛЕ (СНК)

Лабораторная работа №4
«Реализация программы взаимодействия встроенного процессорного ядра с периферийными устройствами СнК»

Авторы:
Любавин Кирилл Дмитриевич
Кузьмин Павел Андреевич

Структура учебного проекта СнК



Конструкции языка СИ

<your_path>/soc_programming/src/rtl/compile/user_programm/main.c

Ключевое слово volatile

Синтаксис конструкции

```
1    volatile declarator ;
```

Пример использования volatile в коде:

```
1    volatile uint32 * statusPtr = 0xF1230000;
```

Volatile — ключевое слово языков C/C++, которое информирует компилятор о том, 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.

Макросы

Макросы используются для определения символьных констант, функций и операций, которые могут быть использованы в программе

```
#define CRC_MODULE_BASE_ADDRESS ( 0x11000 )
```

В данном примере через макрос создаётся константа CRC_MODULE_BASE_ADDRESS, которая соответствует базовому адресу устройства расчёта CRC в системе, который соответствует значению 0x11000.

Указатели

Указатели в языке С используются для работы с памятью напрямую и управлению аппаратными ресурсами. Они позволяют использовать динамическое выделение памяти, создание массивов и структур данных.

```
int* p;   // указатель на целое число
char* s;  // указатель на символ
float* f; // указатель на число с плавающей точкой
```

Пример использования указателя в коде:

```
int* p;   // указатель на целое число
char* s;  // указатель на символ
float* f; // указатель на число с плавающей точкой
```

Конструкции языка СИ

Пример использования конструкции struct в Си

```c
// Подключение библиотеки stdint.h
#include <stdint.h>

// указание базового адреса устройства
#define BASE_ADDRESS ( 0x11000 )

// создание структуры регистровой карты
// apb3 slave устройства
typedef struct {
    uint32_t        crc_data_in;
    uint32_t        crc_data_out;
    uint32_t        control;
} regmap_t;

int main() {

    // создание экземпляра регистровой карты
    // с указанием адреса, где он находится
    volatile regmap_t* regmap = (regmap_t*) BASE_ADDRESS;

    // загрузка исходных данных для расчёта
    regmap->data_in = 0xABCDEF01;

    return 0;
}
```

Конструкция struct

Struct — 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.

stdint.h

stdint.h описывает целочисленные типы данных с установленными диапазонами представления чисел. Вместе с типами данных, в этом файле определены макросы с указанием верхних и нижних границ представляемых значений и макро-функции для формирования диапазонов представляемых значений для каждого типа данных.

Знаковый	Без знаковый	Описание
int_8t	uint_8t	Целочисленные типы данных с шириной диапазона представления чисел 8, 16, 32 и 64 бита, соответственно.
int_16t	uint_16t	
int_32t	uint_32t	
int_64t	uint_64t	

Подготовка к выполнению лабораторной работы

1. Подключить компилятор. С помощью команды gedit ~/.bashrc добавить строчку

```
export PATH=/local/pkims_labs/labs_SoC_PROG/riscv-gcc-10.2.0/bin:/$PATH
```

2. Скопировать проект СнК в свою локальную директорию.
Проект находится по следующему пути:

```
/local/pkims_labs/labs_SoC_PROG/soc_programming
```

3. Скопировать файлы из Л/Р3 в директорию

```
<your_path>/soc_programming/src/rtl
```

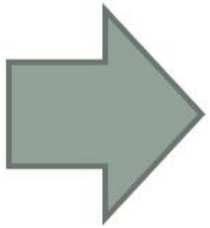
4. Добавить относительные пути до ваших файлов из Л/Р3 в следующий файл

```
<your_path>/soc_programming/src/filelist/main_project.f
```

Подключение блока к топ-уровню СнК

<your_path>/soc_programming/src/rtl/top.sv

```
137
138       //|------------------------
139       //| Student's module place
140       //|------------------------
141
142
143       //| !!!!!!!!!!!!!!!!!!!
144       //| !!!!!!!!!!!!!!!!!!!
145       //| !!!!!!!!!!!!!!!!!!!
146       //| !!!!!!!!!!!!!!!!!!!
147       //| PUT YOUR MODULE HERE
148       //| !!!!!!!!!!!!!!!!!!!
149       //| !!!!!!!!!!!!!!!!!!!
150       //| !!!!!!!!!!!!!!!!!!!
151       //| !!!!!!!!!!!!!!!!!!!
152       //| !!!!!!!!!!!!!!!!!!!
153
```



```
143       APB #(
144           .ADDR_WIDTH         ( APB3_ADDR_WIDTH   ),
145           .DATA_WIDTH         ( APB3_DATA_WIDTH   )
146       ) APB3_sec (
147           .PCLK               ( i_clk             ),
148           .PRESETn            ( i_rst_n           ));
149
150       APB_slave#(
151           .ADDR_WIDTH         ( APB3_ADDR_WIDTH   ),
152           .DATA_WIDTH         ( APB3_DATA_WIDTH   )
153       ) crc_apb_slave (
154           .APB_if             ( APB3_sec.Slave    ));
155
156       always_comb APB3_sec.PADDR      = APB3.PADDR;
157       always_comb APB3_sec.PSEL       = APB3.PSEL;
158       always_comb APB3_sec.PENABLE    = APB3.PENABLE;
159       always_comb APB3_sec.PWRITE     = APB3.PWRITE;
160       always_comb APB3_sec.PWDATA     = APB3.PWDATA;
161
162       always_comb APB3.PREADY         = APB3_sec.PREADY;
163       always_comb APB3.PSLVERR        = APB3_sec.PSLVERR;
164       always_comb APB3.PRDATA         = APB3_sec.PRDATA;
```

Запуск

Подгрузка САПР Synopsys VCS (делается 1 раз для сессии терминала)

```
module load synopsys/VCS/R-2020.12
```

Компиляция С кода

```
cd <your_path>/soc_programming/src/compile/
make clean compile
```

Запуск симуляции в САПР Synopsys VCS

```
cd <your_path>/soc_programming/sim/vcs/
make clean sim
```

Лабораторное задание

1) Подключить блок расчета CRC из лабораторной работы №2 к СнК по APB3 интерфейсу;
2) Написать программу на C, в которой происходит подключение к блоку CRC и считывание флага завершения расчета. Программа должна учитывать все статусные флаги регистра управления из предыдущих Л/Р.
3) Провести функциональную симуляцию проекта; При возникновении ошибок, отладить программу, повторить симуляцию, добиться корректной работы устройства и алгоритма.

! За основу желательно использовать модуль устройства, разработанного в Л/Р №3