ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Журнал практики

Институт № 4 «Радиоэлектроника, инфокоммуникации и информационная безопасность»

Кафедра 402 Учебная группа М4О-303С-21

ФИО обучающегося Соколов Александр Николаевич

Направление подготовки/специальность 11.05.01 Радиоэлектронные системы и комплексы

шифр, наименование направления подготовки/специальности

Вид практики производственная (технологическая)

учебная, производственная, преддипломная или другой вид практики

Оценка за практику _____ Матвеев А.М.

Москва

2024г.

1. Место и сроки проведения практики:

Наименование организации: АО «НТЦ «Модуль»

Сроки проведения практики

дата начала практики: 28.06.2024

дата окончания практики: 25.07.2024

2. Инструктаж по технике безопасности:

_____ / Шкрябай Е.В. / 28 июня 2024г.

подпись проводившего | расшифровка подписи | дата проведения

3. Индивидуальное задание обучающегося: Создание программного продукта, предназначенного для преобразования BSD-файла (JTAG, Boundary Scan) согласно необходимой методике тестирования выбранного устройства в SVF-файл, пригодный для передачи в программу, проводящую граничное тестирование исследуемой платы или её составной части

4. План выполнения индивидуального задания обучающегося:

№ п/п	Место проведения	Тема	Период выполнения
1	АО «НТЦ «Модуль»	1 Инструктаж. Оформление пропусков.	28.06.2024 – 28.06.2024
2	АО «НТЦ «Модуль»	2 Написание программного модуля для анализа и обработки данных из BSD-файла	01.07.2024 – 05.07.2024
3	АО «НТЦ «Модуль»	3 Написание программного модуля, производящего ввод данных для задания параметров проводимого теста	08.07.2024 – 12.07.2024
4	АО «НТЦ «Модуль»	4 Написание программного модуля, производящего вывод итогового SVF-файла	15.07.2024 – 19.07.2024
5	АО «НТЦ «Модуль»	Оформление отчета. Подведение итогов.	22.07.2024 – 24.07.2024

Утверждаю _____ / _____ / 28 июня 2024г.

подпись руководителя от МАИ | расшифровка подписи | дата утверждения

_____ / Шкрябай Е.В / 28 июня 2024г.

подпись руководителя от предприятия | расшифровка подписи | дата утверждения

Ознакомлен _____ / Соколов Л.Н. / 28 июня 2024г.

подпись обучающегося | расшифровка подписи | дата ознакомления

5. Отзыв руководителя практики от организации/предприятия:

Студент Соколов Александр Николаевич группы МЧО-303С-21 проходил практику в подразделении АО „НТЦ „МОДУЛЬ" сектор П-136. В ходе практики необходимо было создать программный продукт, предназначенный для преобразования BSDL-файла согласно заданной методике тестирования выбранного устройства в SVF-файл, предназначенный для передачи в тестовую среду. Была проведена разработка функционального программного модуля и использующих его пользовательских утилит командной строки, выполняющих трансляцию. В ходе практики Александр проявил себя как дисциплинированный сотрудник, умеющий самостоятельно искать информацию, но не боящийся спрашивать совета у коллег. Все задания по практике были выполнены в полном объёме, качественно и в установленный срок. Материалы, изложенные в отчёте обучающегося, полностью соответствуют индивидуальному заданию. Оценка за прохождение практики "отлично".

АКЦИОНЕРНОЕ ОБЩЕСТВО НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР "МОДУЛЬ" ДЛЯ ДОКУМЕНТОВ № 1 ИНН 7714009178

_____ / Шкябай Е.В. / 29.07 2024г.

Подпись руководителя от организации /предприятия | расшифровка подписи | дата

6. Отчет обучающегося по практике:

Введение

Акционерное общество Научно-технический центр «Модуль» более 30 лет успешно работает на российском рынке наукоемких технологий. С 1995 года компания создает высокопроизводительные процессорные ядра и аналогово-цифровые системы-на-кристалле. Сегодня, благодаря высокой квалификации сотрудников и самому современному оснащению, НТЦ «Модуль» разрабатывает и производит аппаратуру управления и контроля самых современных авиационных и космических систем, аппаратно-программные решения в области нейронных сетей, в том числе в части обработки видеопотока и изображений, навигации, связи, обнаружения и распознавания объектов, занимается контрактным выполнением ОКР и НИР.

НТЦ модуль занимается:

– Проектированием и производством вычислительных модулей, систем управления (бортовая и авиационная аппаратура);

– Проектирование интегральных микросхем (услуги микроэлектронного дизайна);

– Внедрение нейронных сетей и отечественной компонентной базы в современные автоматизированные комплексы различных направлений: от навигации до беспилотных автомобилей и робототехники;

– Производство и проектирование систем распознавания и анализа видеоизображений;

– Разработка и производство навигационного оборудования, в том числе высокоточного GNSS позиционирования;

– Разработка СФ-блоков.

В собственности компании вычислительные мощности и технологии, позволяющие проводить наукоемкие исследования и разработки. НТЦ «Модуль» является лицензиатом консорциумов HDMI® и DCP LLC®, имеет аттестованное и оснащенное современным оборудованием сборочное производство, обеспечивающее мелкосерийный выпуск встраиваемых компьютеров и модулей.

Исходные данные

Необходимо разработать программный модуль, предназначенный для преобразования BSD-файла (JTAG, Boundary Scan) согласно необходимой методике тестирования выбранного устройства, сконфигурированной специалистом в SVF-файл, пригодный для передачи в программу, проводящую граничное тестирование исследуемой платы или её составной части, в данном случае OpenOCD.

Описание полученного задания

JTAG (Joint Test Action Group) — это стандартный интерфейс для тестирования и отладки цифровых устройств. Был разработан для упрощения диагностики и проверки сложных электронных систем, таких как интегральные схемы (ИС) и печатные платы (PCB). JTAG обеспечивает средство для взаимодействия с внутренними структурами этих устройств через стандартный физический интерфейс и набор команд. JTAG определяется стандартом IEEE 1149.1, который описывает протокол и электрические характеристики для тестирования и отладки через специализированный интерфейс. На рисунке 1 приведена обобщённая структура микросхемы, оборудованная интерфейсом JTAG.

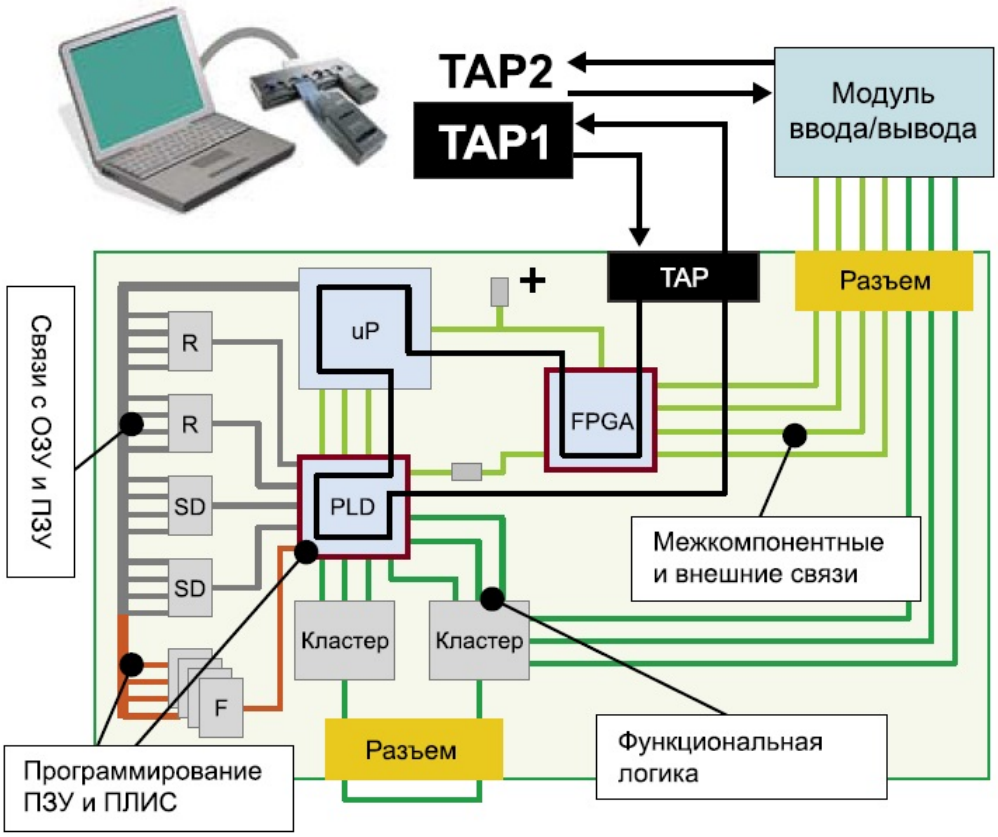

Рисунок 1 – Обобщённая структура микросхемы, оборудованная интерфейсом JTAG

BSD-файла описываем структуру и тип выводов микросхемы, а также описание регистров Boundary Scan предназначенных для проведения тестирования. Пример BSD-файла приведён в листинге 1.

Листинг 1 – Пример BSD-файла

```
 1: entity SIMPLE_IC is
 2:     generic (PHYSICAL_PIN_MAP : string);
 3:     port  (
 4:         LED: out bit_vector(0 to 7);
 5:         BTN: in  bit_vector(0 to 7);
 6:         TMS: in  bit;
 7:         TDI: in  bit;
 8:         TCK: in  bit;
 9:         TDO: out bit;
10:         VDD: linkage bit;
11:         VSS: linkage bit;
12:         NC:  linkage bit_vector(0 to 9)
13:         );
14:
15:     use STD_1149_1_2001.all;
16:
17:     attribute COMPONENT_CONFORMANCE of SIMPLE_IC : entity is
    "STD_1149_1_2001";
18:     attribute PIN_MAP of SIMPLE_IC : entity is PHYSICAL_PIN_MAP;
```

```
19:     constant  PLCC20:PIN_MAP_STRING:=
20:         "LED:     (1,2,3,4), " &
21:         "BTN:     (8,7,6,5), " &
22:         "TMS:     9, " &
23:         "TDI:     10, " &
24:         "TCK:     11, " &
25:         "TDO:     12, " &
26:         "VDD:     13, " &
27:         "VSS:     14, " &
28:         "NC:     (15,16,17,18,19,20) ";
29:
30:     attribute TAP_SCAN_MODE    of TMS : signal is true;
31:     attribute TAP_SCAN_IN        of TDI : signal is true;
32:     attribute TAP_SCAN_CLOCK    of TCK : signal is (10.0e3, BOTH);
33:     attribute TAP_SCAN_OUT        of TDO : signal is true;
34:
35:     attribute INSTRUCTION_LENGTH of SIMPLE_IC : entity is 8;
36:     attribute INSTRUCTION_OPCODE of SIMPLE_IC : entity is
37:         "IDCODE (00000001)," &
38:         "EXTEST (00000010)," &
39:         "BYPASS (11111111)," &
40:         "SAMPLE (00000100) ";
41:     attribute INSTRUCTION_CAPTURE of SIMPLE_IC : entity is "00000001";
42:
43:     attribute IDCODE_REGISTER of SIMPLE_IC : entity is
44:         "0000" &              -- код ревизии или чего-нибудь типа того
45:         "1010101001010101" & -- код модели микросхемы hAA55
46:         "00000000001" &       -- код производителя (соответствует AMD)
47:         "1";                  -- единица по стандарту IEEE1149.1
48:
49:     attribute REGISTER_ACCESS of SIMPLE_IC : entity is
50:         "DEVICE_ID    (IDCODE)," &
51:         "BYPASS    (BYPASS)," &
52:         "BOUNDARY    (EXTEST, SAMPLE)";
53:
54:     attribute BOUNDARY_LENGTH of SIMPLE_IC : entity is 8;
55:     attribute BOUNDARY_REGISTER of SIMPLE_IC : entity is
56:         "0 (BC_1, LED(0), output2, X)," &
57:         "1 (BC_1, LED(1), output2, X)," &
58:         "2 (BC_1, LED(2), output2, X)," &
59:         "3 (BC_1, LED(3), output2, X)," &
60:         "4 (BC_1, BTN(0), input,   X)," &
61:         "5 (BC_1, BTN(1), input,   X)," &
62:         "6 (BC_1, BTN(2), input,   X)," &
63:         "7 (BC_1, BTN(3), input,   X)" ;
64: end SIMPLE_IC;
```

SVF-файл (Serial Vector Format) используется в контексте JTAG (Joint Test Action Group) для описания последовательности команд, которые могут быть использованы для программирования или тестирования цифровых устройств. Формат SVF предоставляет стандартный способ представления тестовых векторов для выполнения различных операций через JTAG-интерфейс. SVF-файл содержит инструкции и данные, которые описывают последовательность команд для взаимодействия с JTAG-совместимыми устройствами. Этот формат позволяет автоматизировать процесс

тестирования и программирования, предоставляя универсальный способ управления устройствами через JTAG. Пример SVF-файла приведён в листинге 2.

Листинг 2 – Пример SVF-файла

```
!Begin Test Program
TRST OFF;
!Disable Test Reset line
ENDIR IDLE;
!End IR scans in IDLE
ENDDR IDLE;
!End DR scans in IDLE
HIR 8 TDI (00);
!8-bit IR header
HDR 16 TDI (FFFF) TDO (FFFF) MASK (FFFF);
!16-bit DR header
TIR 16 TDI (0000);
!16-bit IR trailer
TDR 8 TDI (12);
!16-bit DR trailer
SIR 8 TDI (41);
!8-bit IR scan
SDR 32 TDI (ABCD1234) TDO (11112222);
!32-bit DR scan
STATE DRPAUSE;
!Go to stable state DRPAUSE
RUNTEST 100 TCK ENDSTATE IRPAUSE;
!RUNBIST for 100 TCKs
!End Test Program
```

OpenOCD (Open On-Chip Debugger) – это бесплатный инструмент с открытым исходным кодом для отладки и программирования микроконтроллеров и микропроцессоров через интерфейсы, такие как JTAG и SWD (Serial Wire Debug). OpenOCD поддерживает множество различных архитектур и устройств и часто используется в сочетании с различными IDE (Integrated Development Environment) и инструментами разработки.

Решение поставленной задачи

Модуль работы с BSD-файлом

Листинг кода формирователя итогового SVF-файла приведён на листинге 3, листинге 4, листинге 5.

Листинг 3 – Заголовочный файл библиотеки pininfo.hpp

```
#pragma once
#include <string>
#include <vector>
#include <unordered_map>

// Структура для хранения информации о порте и ячейках

class BsdlPins{
public:

    class PinInfo {
    public:
        enum class StatePin {
            high,
            low,
            z,
            x
        };

        std::string pin;        // номер физического пина, 0 для не выведенных пинов
        std::string label;      // название физического пина
        std::string pin_type;   // тип ячейки in, out, inout
        unsigned int In;        // номер ячейки ввода
        unsigned int Out;       // номер ячейки вывода
        unsigned int Config;    // ячейка управления
        std::string function;   // <function> ячейки BS
        bool turnOff;           // при каком значении в ячейки происходит отключение драйвера 1 или 0

        std::string stateOff;  // состояние выходного отключенного драйвера z, 1 (high), 0 (low)
        std::string safeState; // безопасное значение ячейки X, 1 (high), 0 (low)

    };

    void loadBsdl(std::string filename){

        // Читаем данные из файла и записываем в переменную content
        std::string content = readFile(filename);

        // Получаем даныне о пинах и заносим данные в переменную ping (vector)
        pins = parseBSDFile(content);

        // Получаем данные об имени пина и его номере
        std::unordered_map<std::string, std::string> pinMap = parsePinMap(content);

        // Получаем данные об имени пина и его типе
        std::unordered_map<std::string, std::string> pinTypes = parsePinTypes(content);

        // Устанавливаем связь номеров пинов и их типов
        mapPinNumbersAndTypes(pins, pinMap, pinTypes);

        // Удаляем дублирующиеся пины
        pins = removeDuplicatePins(pins);

    }
    // Функция для вывода информации о пинах
    void printPinInfo(std::ostream &os=std::cout);

protected:
    // Защищенные методы для выполнения операций
```

```
60:
61:     // Функция для чтения файла и возврата его содержимого в виде строки
62:     static std::string readFile(const std::string& filename);
63:
64:     // Функция для преобразования str в bool
65:     static bool stringToBool(const std::string& str);
66:
67:     // Функция для парсинга файла .bsd и извлечения информации о пинах
68:     static std::vector<PinInfo> parseBSDFile(const std::string& content);
69:
70:     // Функция для парсинга строки с описанием пина
71:     static PinInfo parsePinInfo(const std::string& line);
72:
73:     // Функция для парсинга строк с номерами пинов
74:     static std::unordered_map<std::string, std::string> parsePinMap(const std::string&
    content);
75:
76:     // Функция для парсинга строк с типами пинов
77:     static std::unordered_map<std::string, std::string> parsePinTypes(const
    std::string& content);
78:
79:     // Функция для удаления дублирующихся пинов и переноса значения Cell In из
    дубликата в первый пин
80:     static std::vector<PinInfo> removeDuplicatePins(const std::vector<PinInfo>& pins);
81:
82:     // Функция для установки связи номеров пинов и их типов
83:     static void mapPinNumbersAndTypes(std::vector<PinInfo>& pins, const
    std::unordered_map<std::string,
84:         std::string>& pinMap, const std::unordered_map<std::string, std::string>&
    pinTypes);
85:
86: private:
87:     std::vector<PinInfo> pins;
88: };
89:
```

Листинг 4 – Файл реализации библиотеки pininfo.cpp

```cpp
#include <iostream>
#include <fstream>
#include <regex>
#include <string>
#include <unordered_map>

#include "pininfo.hpp"

// Функция для преобразования str в bool
bool BsdlPins::stringToBool(const std::string& str) {
    int value = std::stoi(str);
    return value != 0;
}

// Функция для чтения файла и возврата его содержимого в виде строки
std::string BsdlPins::readFile(const std::string& filename) {
    std::ifstream file(filename);
    if (!file.is_open()) {
        throw std::runtime_error("Could not open file");
    }
    return std::string((std::istreambuf_iterator<char>(file)), std::istreambuf_iterator<char>());
}

// Функция для парсинга строки с описанием пина
BsdlPins::PinInfo BsdlPins::parsePinInfo(const std::string& line) {
    std::regex pinRegex(R"(\s*(\d+)\s*\((\w+),\s*(\S*),\s*(\w+),?\s*(\w*),?\s*(\S*),?\s*(\w*).\s*(\w*))");
    std::smatch match;

    PinInfo pinInfo;
    if (std::regex_search(line, match, pinRegex)) {
        pinInfo.label = match[3].str();
        pinInfo.function = match[4].str();
        pinInfo.safeState = match[5].str();
        match[7].str().empty() ? pinInfo.turnOff = 0 : pinInfo.turnOff = stringToBool(match[7].str());
        pinInfo.stateOff = match[8].str();

        if (match[4].str() == "INPUT") {
            pinInfo.In = std::stoi(match[1].str());
            match[6].str().empty() ?  pinInfo.Config = 0 : pinInfo.Config = std::stoi(match[6].str());
            pinInfo.Out = 0;
        } else if (match[4].str() == "OUTPUT3") {
            pinInfo.Out = std::stoi(match[1].str());
            match[6].str().empty() ? pinInfo.Config = 0 : pinInfo.Config = std::stoi(match[6].str());
            pinInfo.In = 0;
        } else {
            pinInfo.In = 0; pinInfo.Out = 0; pinInfo.Config = 0;
        }
    }

    return pinInfo;
}

// Функция для парсинга строк с номерами пинов
std::unordered_map<std::string, std::string> BsdlPins::parsePinMap(const std::string& content) {
    std::unordered_map<std::string, std::string> pinMap;
    std::regex pinEntryRegex(R"(\s*(\w+)\s*:\s*(\d+),)");
```

```
57:     auto lines_begin = std::sregex_iterator(content.begin(), content.end(),
    pinEntryRegex);
58:     auto lines_end = std::sregex_iterator();
59:
60:     for (std::sregex_iterator i = lines_begin; i != lines_end; ++i) {
61:         std::smatch match = *i;
62:         std::string label = match[1].str();
63:         std::string pin = match[2].str();
64:         pinMap[label] = pin;
65:     }
66:
67:     return pinMap;
68: }
69:
70: // Функция для парсинга строк с типами пинов
71: std::unordered_map<std::string, std::string> BsdlPins::parsePinTypes(const
    std::string& content) {
72:     std::unordered_map<std::string, std::string> pinTypes;
73:     std::regex pinTypeRegex(R"(\s*(\w+)\s*:\s*(\w+)\s*(\w*);)");
74:     auto lines_begin = std::sregex_iterator(content.begin(), content.end(),
    pinTypeRegex);
75:     auto lines_end = std::sregex_iterator();
76:
77:     for (std::sregex_iterator i = lines_begin; i != lines_end; ++i) {
78:         std::smatch match = *i;
79:         std::string label = match[1].str();
80:         std::string pin_type = match[2].str();
81:         pinTypes[label] = pin_type;
82:     }
83:
84:     return pinTypes;
85: }
86:
87: // Функция для парсинга файла .bsd и извлечения информации о пинах
88: std::vector<BsdlPins::PinInfo> BsdlPins::parseBSDFile(const std::string&
    content) {
89:     std::vector<PinInfo> pins;
90:     std::regex lineRegex(R"((\d+\s*\(BC_\d+,.*\)))");
91:     auto lines_begin = std::sregex_iterator(content.begin(), content.end(),
    lineRegex);
92:     auto lines_end = std::sregex_iterator();
93:
94:     for (std::sregex_iterator i = lines_begin; i != lines_end; ++i) {
95:         std::smatch match = *i;
96:         PinInfo pin = parsePinInfo(match.str());
97:         pins.push_back(pin);
98:     }
99:
100:     return pins;
101: }
102:
103: // Функция для удаления дублирующихся пинов и переноса значения Cell In из
    дубликата в первый пин
104: std::vector<BsdlPins::PinInfo> BsdlPins::removeDuplicatePins(const
    std::vector<PinInfo>& pins) {
105:     std::unordered_map<std::string, PinInfo> pinMap;
106:     for (const auto& pin : pins) {
107:         if (pinMap.find(pin.pin) == pinMap.end()) {
108:             pinMap[pin.pin] = pin;
109:         } else {
110:             if (pin.function == "INPUT" && pinMap[pin.pin].function ==
    "OUTPUT3") {
111:                 pinMap[pin.pin].In = pin.In;
112:             }
113:         }
114:     }
```

```
115:
116:     std::vector<PinInfo> result;
117:     for (const auto& pair : pinMap) {
118:         result.push_back(pair.second);
119:     }
120:
121:     return result;
122: }
123:
124: // Функция для установки связи номеров пинов и их типов
125: void BsdlPins::mapPinNumbersAndTypes(std::vector<PinInfo>& pins, const
    std::unordered_map<std::string,
126:     std::string>& pinMap, const std::unordered_map<std::string, std::string>&
    pinTypes) {
127:     for (auto& pin : pins) {
128:         if (pinMap.find(pin.label) != pinMap.end()) {
129:             pin.pin = pinMap.at(pin.label);
130:         } else {
131:             pin.pin = "0";  // Используем 0 для ячеек BS, к которым не привязаны
    пины
132:         }
133:
134:         if (pinTypes.find(pin.label) != pinTypes.end()) {
135:             pin.pin_type = pinTypes.at(pin.label);
136:         } else {
137:             pin.pin_type = "unknown";  // Используем "unknown" для неизвестных
    типов
138:         }
139:     }
140: }
141:
142:
143: // Функция для вывода информации о пинах
144: void BsdlPins::printPinInfo(std::ostream &os) {
145:     for (const auto& pin : pins) {
146:         os << "Pin: " << pin.pin
147:             << ", Port Name: " << (pin.label.empty() ? "*" : pin.label) //
    condition ? true_value : false_value
148:             << ", Pin type: " << pin.pin_type
149:             << ", Function: " << pin.function
150:             << ", Cell In: " << pin.In
151:             << ", Cell Out: " << pin.Out
152:             << ", Cell Config: " << pin.Config
153:             << ", Disable Value: " << pin.turnOff
154:             << ", Safe State: " << (pin.safeState.empty() ? "N/A" :
    pin.safeState)
155:             << ", State Off: " << (pin.stateOff.empty() ? "N/A" : pin.stateOff)
156:             << std::endl;
157:     }
158: }
```

Листинг 5 – Файл, работающий с библиотекой pininfo.hpp

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <vector>
#include <string>

# include "pininfo.hpp"

int main(int argc, char* argv[]) {
    if (argc < 2) {
        std::cerr << "Usage: " << argv[0] << " <bsd-file> " << std::endl;
        return 1;
    }

    // Получаем имя файла
    std::string filename = argv[1];

    BsdlPins BsdlPins;

    // Вызываем методы загрузки и обработки bsd
    BsdlPins.loadBsdl(filename);

    // Выводим информацию о пинах
    BsdlPins.printPinInfo();

    return 0;
}
```

Модуль формирования задачи тестирования

Листинг кода формирователя задачи тестирования приведён на листинге 6, листинге 7, листинге 8.

Листинг 6 – Заголовочный файл библиотеки svf-calc.hpp

```
 1: #pragma once
 2: #include <string>
 3: #include <vector>
 4: #include <nlohmann/json.hpp>
 5:
 6: using json = nlohmann::json;
 7:
 8: class JsonForm {
 9: public:
10:     void fileForm(int argc, char *argv[]);
11:
12: private:
13:     std::vector<std::vector<std::string>> pinsList;
14:     std::vector<std::vector<std::string>> writeStatusList;
15:     std::vector<std::vector<std::string>> readStatusList;
16:     std::string filenameBSD;
17:
18:     void parsingArguments(int argc, char *argv[],
    std::vector<std::vector<std::string>> &pinsList,
19:                           std::vector<std::vector<std::string>>
    &writeStatusList, std::vector<std::vector<std::string>> &readStatusList,
20:                           std::string &filenameBSD);
21:
22:     bool writeJsonToFile(const std::string& filename, const nlohmann::json&
    jsonObject);
23:
24:     std::string replaceExtension(const std::string& filename, const std::string&
    oldExt, const std::string& newExt);
25:
26:     json createJsonObject(const std::vector<std::vector<std::string>>& pinsList,
27:                           const std::vector<std::vector<std::string>>&
    writeStatusList,
28:                           const std::vector<std::vector<std::string>>&
    readStatusList);
29: };
```

Листинг 7 – Файл реализации библиотеки svf-calc.hpp

```
#include <iostream>
#include <fstream>
#include "svf-calc.hpp"

void JsonForm::fileForm(int argc, char *argv[]){
    // Парсинг аргументов
    parsingArguments(argc, argv, pinsList, writeStatusList, readStatusList, filenameBSD);

    // Создание имени файла json
    std::string filename = replaceExtension(filenameBSD, ".bsd", "_test.json");

    // Создание json объекта
    json jsonArray = createJsonObject(pinsList, writeStatusList, readStatusList);

    // Запись json в файл
    if (writeJsonToFile(filename, jsonArray)) {
        std::cout << "Создан файл: " << filename << std::endl;
    }
    return;
}

// Функция для парсинга аргументов командной строки
void JsonForm::parsingArguments(int argc, char *argv[], std::vector<std::vector<std::string>> &pinsList,
                    std::vector<std::vector<std::string>> &writeStatusList, std::vector<std::vector<std::string>> &readStatusList,
                    std::string &filenameBSD) {
    std::vector<std::string> pins;
    std::vector<std::string> writeStatus;
    std::vector<std::string> readStatus;

    for (int i = 1; i < argc; ++i) {
        if (std::string(argv[i]) == "--pins" && i + 1 < argc) {
            if (!pins.empty()) {
                pinsList.push_back(pins);
                writeStatusList.push_back(writeStatus);
                readStatusList.push_back(readStatus);
                pins.clear();
                writeStatus.clear();
                readStatus.clear();
            }
            std::string pinsStr = argv[++i];
            size_t pos = 0;
            while ((pos = pinsStr.find(',')) != std::string::npos) {
                pins.push_back(pinsStr.substr(0, pos));
                pinsStr.erase(0, pos + 1);
            }
            pins.push_back(pinsStr);
        } else if (std::string(argv[i]) == "--write" && i + 1 < argc) {
            std::string writeStr = argv[++i];
            size_t pos = 0;
            while ((pos = writeStr.find(',')) != std::string::npos) {
                writeStatus.push_back(writeStr.substr(0, pos));
                writeStr.erase(0, pos + 1);
            }
            writeStatus.push_back(writeStr);
        } else if (std::string(argv[i]) == "--read" && i + 1 < argc) {
            std::string readStr = argv[++i];
            size_t pos = 0;
            while ((pos = readStr.find(',')) != std::string::npos) {
                readStatus.push_back(readStr.substr(0, pos));
                readStr.erase(0, pos + 1);
```

```
            }
            readStatus.push_back(readStr);
        } else if (std::string(argv[i]) == "--filename" && i + 1 < argc) {
            filenameBSD = argv[++i];
        }
    }
    if (!pins.empty()) {
        pinsList.push_back(pins);
        writeStatusList.push_back(writeStatus);
        readStatusList.push_back(readStatus);
    }
}

// Функция создания и записи в файл
bool JsonForm::writeJsonToFile(const std::string& filename, const
nlohmann::json& jsonObject) {
    std::ofstream file(filename);
    if (!file.is_open()) {
        std::cerr << "Файл: " << filename << " не может быть открыт" <<
std::endl;
        return false;
    }
    file << jsonObject.dump(4); // параметр 4 задает отступы для красивого
форматирования
    file.close();
    return true;
}

// Функция замены расширения файла
std::string JsonForm::replaceExtension(const std::string& filename, const
std::string& oldExt, const std::string& newExt) {
    size_t pos = filename.rfind(oldExt);
    if (pos != std::string::npos && pos == filename.length() - oldExt.length())
{
        return filename.substr(0, pos) + newExt;
    } else {
        std::cerr << "Некорректное расширение файла: " << filename << std::endl;
        exit(1);
    }
}

// Функция создания json объекта
json JsonForm::createJsonObject(const std::vector<std::vector<std::string>>&
pinsList,
                      const std::vector<std::vector<std::string>>&
writeStatusList,
                      const std::vector<std::vector<std::string>>& readStatusList)
{
    json jsonArray = json::array();
    for (size_t i = 0; i < pinsList.size(); ++i) {
        json jsonObject;
        jsonObject["pins"] = pinsList[i];
        jsonObject["write"] = writeStatusList[i];
        jsonObject["read"] = readStatusList[i];
        jsonArray.push_back(jsonObject);
    }
    return jsonArray;
}
```

Листинг 8 – Файл, работающий с библиотекой svf-calc.hpp

```
#include <iostream>
#include "svf-calc.hpp"

using json = nlohmann::json;
```

```
 5:
 6:  // Основная функция
 7:  int main(int argc, char *argv[]) {
 8:      if (argc < 2) {
 9:          std::cerr << "Общая структура: --filename \"name_file_BSD.bsd\" --pins
     \"pin_name_1, ..., pin_name_n\" --write \"pin_status_1, ..., pin_status_n\" --
     read \"pin_status_1, ..., pin_status_n\"" << std::endl;
10:          return 1;
11:      } else if (argc < 4) {
12:          std::cerr << "Структура пинов: --pins \"pin_name_1, ..., pin_name_n\" --
     write \"pin_status_1, ..., pin_status_n\" --read \"pin_status_1, ...,
     pin_status_n\"" << std::endl;
13:          return 1;
14:      }
15:
16:      JsonForm JsonForm;
17:
18:      JsonForm.fileForm(argc, argv);
19:
20:      return 0;
21:  }
```

Модуль формирования итогового SVF-файла

Листинг кода формирователя итогового SVF-файла приведён на листинге 9, листинге 10, листинге 11.

Листинг 9 – Заголовочный файл библиотеки svf-generator.hpp

```
 1:  #pragma once
 2:  #include <string>
 3:  #include <unordered_set>
 4:
 5:  class StateArg {
 6:  public:
 7:      // Публичный метод для запуска формирования SVF
 8:      void formation_svf(int argc, char *argv[]);
 9:
10:  private:
11:      // Параметры для формирования SVF
12:      std::string filename_bsd = "NO FILE";
13:      std::string filename_json = "NO FILE";
14:      std::string trst_state = "OFF";
15:      std::string endir_state = "IDLE";
16:      std::string enddr_state = "IDLE";
17:
18:      // Приватные методы
19:      void print_usage();
20:      bool is_valid_state(const std::string& state, const std::string valid_states[], size_t
     count);
21:      bool has_extension(const std::string& filename, const std::unordered_set<std::string>&
     validExtensions);
22:      void parse_arguments(int argc, char *argv[]);
23:      std::string replaceExtension(const std::string& filename, const std::string& oldExt,
     const std::string& newExt);
24:      void createFile(char *argv[]);
25:  };
```

Листинг 10 – Файл реализации библиотеки svf-generator.hpp

```
#include "svf-generator.hpp"
#include <iostream>
#include <fstream>
#include <vector>
#include <unordered_set>
#include <getopt.h>

// Публичная метод запускающий privat методы
void StateArg::formation_svf(int argc, char *argv[]){
    // Парсинг аргументов
    parse_arguments(argc, argv);

    // Создание файла svf
    createFile(argv);
}

// Функция вывода help
void StateArg::print_usage() {
    std::cout << "Usage: program [options]\n"
              << "Options:\n"
              << "  -b, --bsdl    Add a BSDL-file\n"
              << "  -j, --json    Add a JSON-file\n"
              << "  -t, --trst    Set the TRST state (ON, OFF, z, ABSENT)\n"
              << "  -i, --endir   Set the ENDIR state (IRPAUSE, DRPAUSE, RESET, IDLE)\n"
              << "  -d, --enddr   Set the ENDDR state (IRPAUSE, DRPAUSE, RESET, IDLE)\n"
              << "  -h, --help    Show this help message\n";
}

// Функция проверки корректности id введённого в CLI аргументов
bool StateArg::is_valid_state(const std::string& state, const std::string valid_states[], size_t count) {
    for (size_t i = 0; i < count; ++i) {
        if (state == valid_states[i]) {
            return true;
        }
    }
    return false;
}

// Функция проверки расширения файла
bool StateArg::has_extension(const std::string& filename, const std::unordered_set<std::string>& validExtensions) {
    size_t pos = filename.rfind('.');
    if (pos != std::string::npos && pos != filename.length() - 1) {
        std::string fileExt = filename.substr(pos + 1);
        return validExtensions.find(fileExt) != validExtensions.end();
    }
    return false;
}

// Функция парсинга аргументов и вывода записанных аргументов в консоль
void StateArg::parse_arguments(int argc, char *argv[]){

    // Инициализация вспомагательных переменных
    int option_index = 0;
    int c;

    // Инициализация доступных аргументов
    static struct option long_options[] = {
        {"bsdl", required_argument, 0, 'b'},
        {"json", required_argument, 0, 'j'},
        {"trst", required_argument, 0, 't'},
```

```
61:         {"endir", required_argument, 0, 'i'},
62:         {"enddr", required_argument, 0, 'd'},
63:         {"help", no_argument, 0, 'h'},
64:         {0, 0, 0, 0}
65:     };
66:
67:     // Инициализация доступных состояний аргументов
68:     const std::string trst_states[] = {"ON", "OFF", "z", "ABSENT"};
69:     const std::string endir_enddr_states[] = {"IRPAUSE", "DRPAUSE", "RESET",
    "IDLE"};
70:
71:     // Цикл проверки всех переданных аргументов
72:     while ((c = getopt_long(argc, argv, "b:j:t:i:d:h", long_options,
    &option_index)) != -1) {
73:         switch (c) {
74:             case 'b':
75:                 filename_bsd = optarg;
76:                 if (!has_extension(filename_bsd,{"bsd","bsdl"})) {
77:                     std::cerr << "Неверное расширение BSDL-файла (.bsd)\n";
78:                     abort();
79:                 }
80:                 break;
81:             case 'j':
82:                 filename_json = optarg;
83:                 if (!has_extension(filename_json, {"json"})) {
84:                     std::cerr << "Неверное расширение JSON-файла (.json)\n";
85:                     abort();
86:                 }
87:                 break;
88:             case 't':
89:                 trst_state = optarg;
90:                 if (!is_valid_state(trst_state, trst_states, 4)) {
91:                     std::cerr << "Неверное состояние --trst. Возможные состояния
    ON, OFF, z, or ABSENT.\n";
92:                     abort();
93:                 }
94:                 break;
95:             case 'i':
96:                 endir_state = optarg;
97:                 if (!is_valid_state(endir_state, endir_enddr_states, 4)) {
98:                     std::cerr << "Неверное состояние --endir. Возможные
    состояния IRPAUSE, DRPAUSE, RESET, or IDLE.\n";
99:                     abort();
100:                }
101:                break;
102:            case 'd':
103:                enddr_state = optarg;
104:                if (!is_valid_state(enddr_state, endir_enddr_states, 4)) {
105:                    std::cerr << "Неверное состояние --enddr. Возможные
    состояния IRPAUSE, DRPAUSE, RESET, or IDLE.\n";
106:                    abort();
107:                }
108:                break;
109:            case 'h':
110:                print_usage();
111:                return;
112:            default:
113:                abort();
114:        }
115:    }
116:
117:    // Проверка на наличие BSD-файла и JSON-файла
118:    if((filename_bsd == "NO FILE")||(filename_json == "NO FILE")){
119:        std::cout << "Необходимо указать имя BSDL-файла и JSON-файла" <<
    std::endl;
120:        abort();
```

```
121:     }
122: 
123:     // Вывод записанных аргументов
124:     std::cout << "\nBSDL-file: " << filename_bsd << "\n";
125:     std::cout << "JSON-file: " << filename_json << "\n";
126:     std::cout << "TRST state: " << trst_state << "\n";
127:     std::cout << "ENDIR state: " << endir_state << "\n";
128:     std::cout << "ENDDR state: " << enddr_state << "\n";
129: }
130: 
131: // Функция замены расширения файла
132: std::string StateArg::replaceExtension(const std::string& filename, const
    std::string& oldExt, const std::string& newExt) {
133:     size_t pos = filename.rfind(oldExt);
134:     if (pos != std::string::npos && pos == filename.length() - oldExt.length())
    {
135:         return filename.substr(0, pos) + newExt;
136:     } else {
137:         std::cerr << "Некорретное расширение файла: " << filename << std::endl;
138:         exit(1);
139:     }
140: }
141: 
142: // Функция создающая файл и заполняющая его в соотвествии с json
143: void StateArg::createFile(char *argv[]){
144:     // Создание имени файла с расширением svf
145:     std::string filename_svf = replaceExtension(filename_json, ".json", ".svf");
146: 
147:     // Открытие файла для записи
148:     std::ofstream svfFile(filename_svf);
149: 
150:     // Запись данных в файл (длина регистра 5 bit для STM32F1)
151:     svfFile << "! Начать программу тестирования\n"
152:                "TRST " << trst_state << ";\n";
153: 
154:     svfFile << "ENDIR "<< endir_state << ";\n";
155: 
156:     svfFile << "ENDDR "<< enddr_state << ";\n";
157: 
158:     unsigned int count = 1;
159: 
160:     unsigned int instr_len = 5;
161: 
162:     std::string EXTEST = "00000";
163: 
164:     unsigned int bound_len = 139;
165: 
166:     std::vector<char> pin_tdi(bound_len, 0);
167: 
168:     std::vector<char> pin_tdo(bound_len, 0);
169: 
170:     std::vector<char> pin_mask(bound_len, 0);
171: 
172:     for(int i = 0; i < count; i++){
173:         svfFile << "TIR " << instr_len << " TDI (" << EXTEST << ")\n";
174: 
175:         // svfFile << "SDR " << bound_len << " TDI " << pin_tdi << " TDO " <<
    pin_tdo << " MASK (" << pin_mask << ");\n";
176: 
177:         svfFile << "RUNTEST 100 TCK ENDSTATE IDLE;\n";
178:     }
179: 
180:     // Закрытие файла
181:     svfFile.close();
182: 
183:     // Успешное завершение программы
```

```
184:     std::cout << "\nФайл " << filename_svf << " успешно создан." << std::endl;
185: }
```

Листинг 11 – Файл, работающий с библиотекой svf-generator.hpp

```
 1: #include <iostream>
 2: #include "svf-generator.hpp"
 3: #include "pininfo.hpp"
 4:
 5: int main(int argc, char* argv[]) {
 6:
 7:     // Получаем имя файла
 8:     std::string filename = argv[1];
 9:
10:     BsdlPins BsdlPins;
11:
12:     // Вызываем методы загрузки и обработки bsd
13:     BsdlPins.loadBsdl(filename);
14:
15:     StateArg StateArg;
16:
17:     // Парсим аргументы введённые при запуске и создаём SVF-файл
18:     StateArg.formation_svf(argc, argv);
19:
20:     return 0;
21: }
```

Список литературы

1. Интерфейс JTAG? — Это очень просто URL: https://habr.com/ru/articles/190012/;
2. Разглядывая JTAG: *.bsdl своими руками URL: https://habr.com/ru/articles/660795/;
3. IEEE Std. 1149.1 - Standard Test Access Port and Boundary-Scan Architecture URL: https://grouper.ieee.org/groups/1149/1/;
4. JTAG-тест: мифы и реальность URL: https://tech-e.ru/2011_6_57.php;
5. IEEE Std 1149.1 (JTAG) Testability URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ti.com/lit/an/ssya002c/ssya002c.pdf;
6. SVF File URL: https://www.xjtag.com/about-jtag/svf-files/;
7. SVF SERIAL VECTOR FORMAT SPECIFICATION JTAG | BOUNDARY SCAN URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.asset-intertech.com/wp-content/uploads/2020/09/svf-serial-vector-format-specification-jtag-boundary-scan-revision-e.pdf.

_____/ Соколов Д. К. /_ 25 июля 2024г.

подпись обучающегося расшифровка подписи дата