Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной NIOSII\_1**

Дисциплина: Автоматизация проектирования дискретных устройств (на английском языке)

Выполнил студент гр. 5130901/10101 М.Т. Непомнящий

(подпись)

Руководитель А.А. Федотов

(подпись)

Санкт-Петербург

2024

**Оглавление**

[1. Задание 5](#_Toc164595254)

[1.1. Цель работы 5](#_Toc164595255)

[1.2. Структура проекта 5](#_Toc164595256)

[2. Ход работы 6](#_Toc164595257)

[2.1. Создание проекта 6](#_Toc164595258)

[Начало работы в PD 6](#_Toc164595259)

[2.2. Настройка сигналов 6](#_Toc164595260)

[Настройка clk 6](#_Toc164595261)

[2.3. Подключение сигналов 7](#_Toc164595262)

[Настройка и подключение On-Chip Memory (RAM and ROM) 7](#_Toc164595263)

[Настройка и подключение Nios II Processor 8](#_Toc164595264)

[Настройка и подключение PIO (Parallel I/O) 9](#_Toc164595265)

[Адресация системы 10](#_Toc164595266)

[2.4. Анализ полученной системы 11](#_Toc164595267)

[Анализ предустановок 11](#_Toc164595268)

[Проверка блока 12](#_Toc164595269)

[Проверка отсутствия проблемных подключений 12](#_Toc164595270)

[Анализ с помощью Schematic 13](#_Toc164595271)

[Генерация системы 13](#_Toc164595272)

[2.5. Подключение файлов к проекту 15](#_Toc164595273)

[2.6. Анализ проекта 16](#_Toc164595274)

[RTL Viewer 16](#_Toc164595275)

[Назначение выходов проекта 16](#_Toc164595276)

[2.7. Создание программной части проекта 17](#_Toc164595277)

[Создание файла Source 17](#_Toc164595278)

[Компиляция проекта в Eclipse 18](#_Toc164595279)

[Полная компиляция проекта в QP 19](#_Toc164595280)

[2.8. Реализация на плате 20](#_Toc164595281)

[3. Дополнительные задания 21](#_Toc164595282)

[3.1. Дополнительное задание 1 21](#_Toc164595283)

[Создание программного файла Lab1\_task1\_source.c 21](#_Toc164595284)

[Реализация на плате 21](#_Toc164595285)

[3.2. Дополнительное задание 2 22](#_Toc164595286)

[Увеличение разрядности pio\_SW 22](#_Toc164595287)

[Создание программного файла Lab1\_task2\_source.c 22](#_Toc164595288)

[Реализация на плате 23](#_Toc164595289)

[4. Вывод 24](#_Toc164595290)

**Список иллюстраций**

[Рис. 1 – Структура проекта 5](#_Toc164595208)

[Рис. 2 – Создание проекта 6](#_Toc164595209)

[Рис. 3 – Исходное окно PD 6](#_Toc164595210)

[Рис. 4 – Настройка компонента clk 6](#_Toc164595211)

[Рис. 5 – Настройка модуля onchip \_mem 7](#_Toc164595212)

[Рис. 6 – Подключение onchip\_mem 7](#_Toc164595213)

[Рис. 7 – Настройка модуля nios2\_PD 8](#_Toc164595214)

[Рис. 8 – Подключение модуля nios2\_PD 8](#_Toc164595215)

[Рис. 9 – Настройка параметров exception в модуле nios2\_PD 9](#_Toc164595216)

[Рис. 10 - Настройка модуля pio\_LED 9](#_Toc164595217)

[Рис. 11 – Подключение модуля pio\_LED 9](#_Toc164595218)

[Рис. 12 – Настройка модуля pio\_SW 10](#_Toc164595219)

[Рис. 13 – Подключение модуля pio\_SW 10](#_Toc164595220)

[Рис. 14 – Окно Address Map после выполнения автоматической адресации 10](#_Toc164595221)

[Рис. 15 – Внешний вид созданной системы 11](#_Toc164595222)

[Рис. 16 – Предустановки системы 11](#_Toc164595223)

[Рис. 17 – Окно Messages с предупреждением 12](#_Toc164595224)

[Рис. 18 – Символ системы 12](#_Toc164595225)

[Рис. 19 – Анализ проблемных подключений 12](#_Toc164595226)

[Рис. 20 – Show System with QSYS Interconnect 13](#_Toc164595227)

[Рис. 21 – Schematic 13](#_Toc164595228)

[Рис. 22 – Предустановки окна Genreration (по умолчанию) 14](#_Toc164595229)

[Рис. 23 – Проверка генерации HDL 14](#_Toc164595230)

[Рис. 24 – Подключение файла .qip к проекту 15](#_Toc164595231)

[Рис. 25 – Синтаксис файла Lab1.sv 15](#_Toc164595232)

[Рис. 26 – Создание символа файла верхнего уровня 15](#_Toc164595233)

[Рис. 27 – Выбор файла верхнего уровня 16](#_Toc164595234)

[Рис. 28 – Схема проекта в RTL Viewer 16](#_Toc164595235)

[Рис. 29 – Назначение выводов платы средствами Pin Planner 16](#_Toc164595236)

[Рис. 30 – Unused Pins 17](#_Toc164595237)

[Рис. 31 – Создание проекта в Eclipse 17](#_Toc164595238)

[Рис. 32 – Пред настройки Source File’а 17](#_Toc164595239)

[Рис. 33 – Синтаксис файла Lab1\_source.c 18](#_Toc164595240)

[Рис. 34 – Build Project 18](#_Toc164595241)

[Рис. 35 – Настройка BSP Editor 18](#_Toc164595242)

[Рис. 36 – Build Project после изменения настроек BSP Editor 19](#_Toc164595243)

[Рис. 37 – Окно Make Targets 19](#_Toc164595244)

[Рис. 38 – Проверка размера и базового адреса памяти 19](#_Toc164595245)

[Рис. 39 – Добавление meminit.qip в проект 19](#_Toc164595246)

[Рис. 40 – SDC файл 20](#_Toc164595247)

[Рис. 41 – Добавление SDC файла 20](#_Toc164595248)

[Рис. 42 – Временные характеристики устройства 20](#_Toc164595249)

[Рис. 43 – Изменённый исходный файл 21](#_Toc164595250)

[Рис. 44 – Изменение настроек модуля pio\_SW 22](#_Toc164595251)

[Рис. 45 – Подключение SW[7:0] (Pin Planner) 22](#_Toc164595252)

[Рис. 46 – Программный файл с реализацией сложения по модулю SW[7:0] 23](#_Toc164595253)

# Задание

## Цель работы

Познакомиться с процедурой реализации «системы на кристалле» – проекта на базе процессора NIOSII, включая следующие этапы:

✓ Создание проекта в пакете Quartus Prime (QP)

✓ Создание аппаратной части проекта помощью приложения Platform Designer (PD)

✓ Создание программной части проекта в рамках оболочки NIOSII IDE

✓ Проверка работы проекта на плате

## Структура проекта

Процессор NIOSII на светодиодах LED8 … LED1 отображает двоичные коды чисел от 0 до 255, под управлением данных, получаемых с переключателей SW:

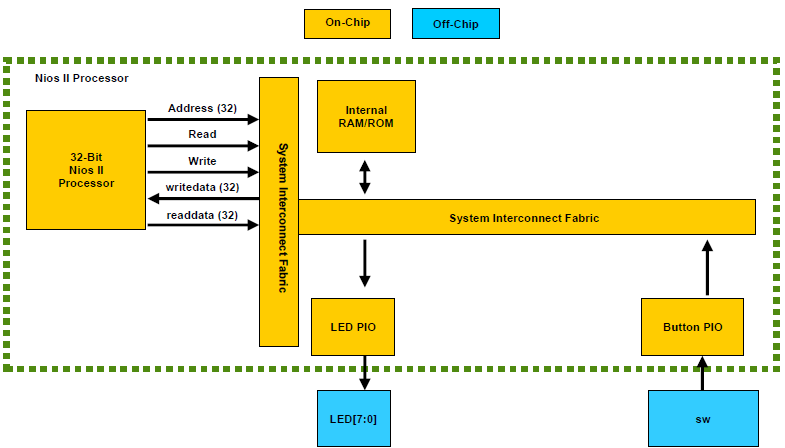


Рис. 1 – Структура проекта

Устройство, которое содержит master и 3 slave: 2 модуля my\_slave и 1 модуль my\_Dslave (default slave)

Master получает некоторые данные через Conduit, через 8-разрядный интерфейс мастер осуществляет адресный доступ к одному из slave’ов, настраивает соответственно slave’s, либо что-то в них записывает, каждый из slave’ов имеет в себе Conduit, который помогает посмотреть на выводе slave’а то, что мы туда записали из мастера.

# Ход работы

## Создание проекта

Создадим проект, указав параметры, представленные на Рис. 2 ниже:

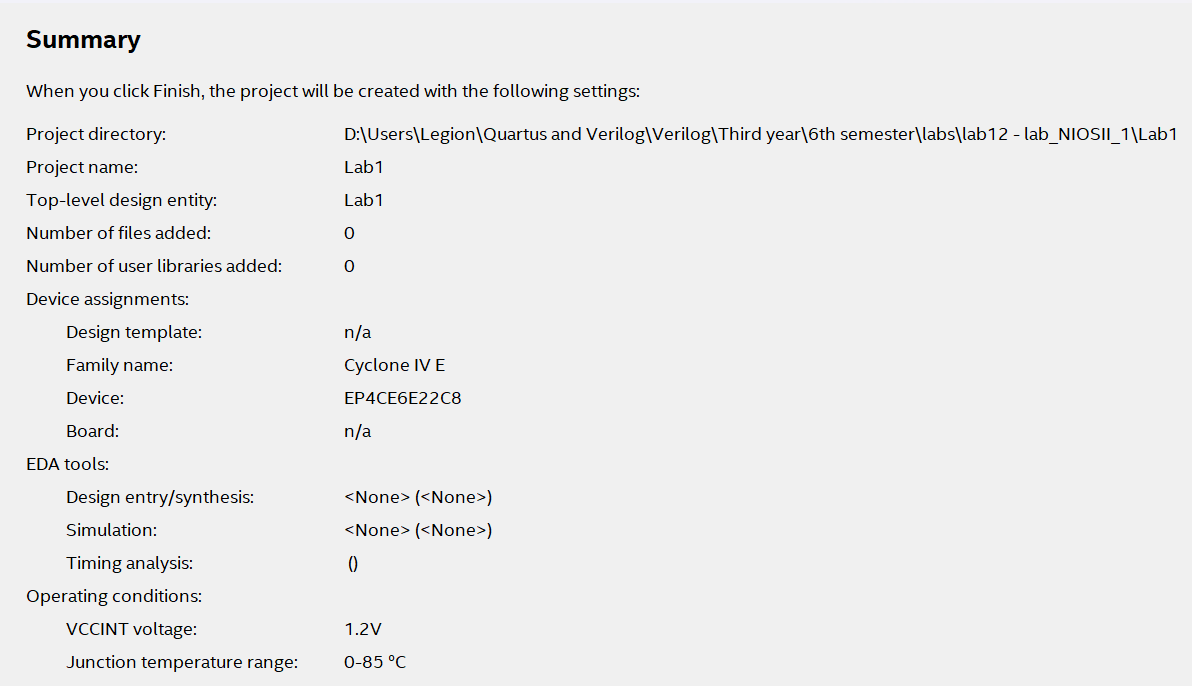


Рис. 2 – Создание проекта

### Начало работы в PD

Откроем PD и сохраним систему:

Изображение выглядит как текст, программное обеспечение, число, Шрифт

Автоматически созданное описание

Рис. 3 – Исходное окно PD

## Настройка сигналов

### Настройка clk

Зададим значение Clock frequency = 25 MHz (частота кварцевого генератора на плате miniDiLaB-CIV, с которой предстоит работать), а также Reset synchronous edges = Deassert:

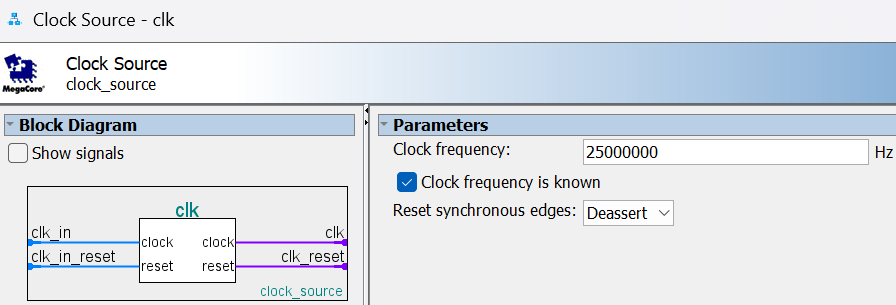


Рис. 4 – Настройка компонента clk

## Подключение сигналов

### Настройка и подключение On-Chip Memory (RAM and ROM)

Через окно Library добавим в проект модуль On-Chip Memory (RAM and ROM), переименуем его на onchip \_mem и зададим следующие параметры:

* Тип памяти – RAM
* Размер памяти – 16384 байт (16 Кб)
* Остальные галочки и настройки так, как показано на ниже

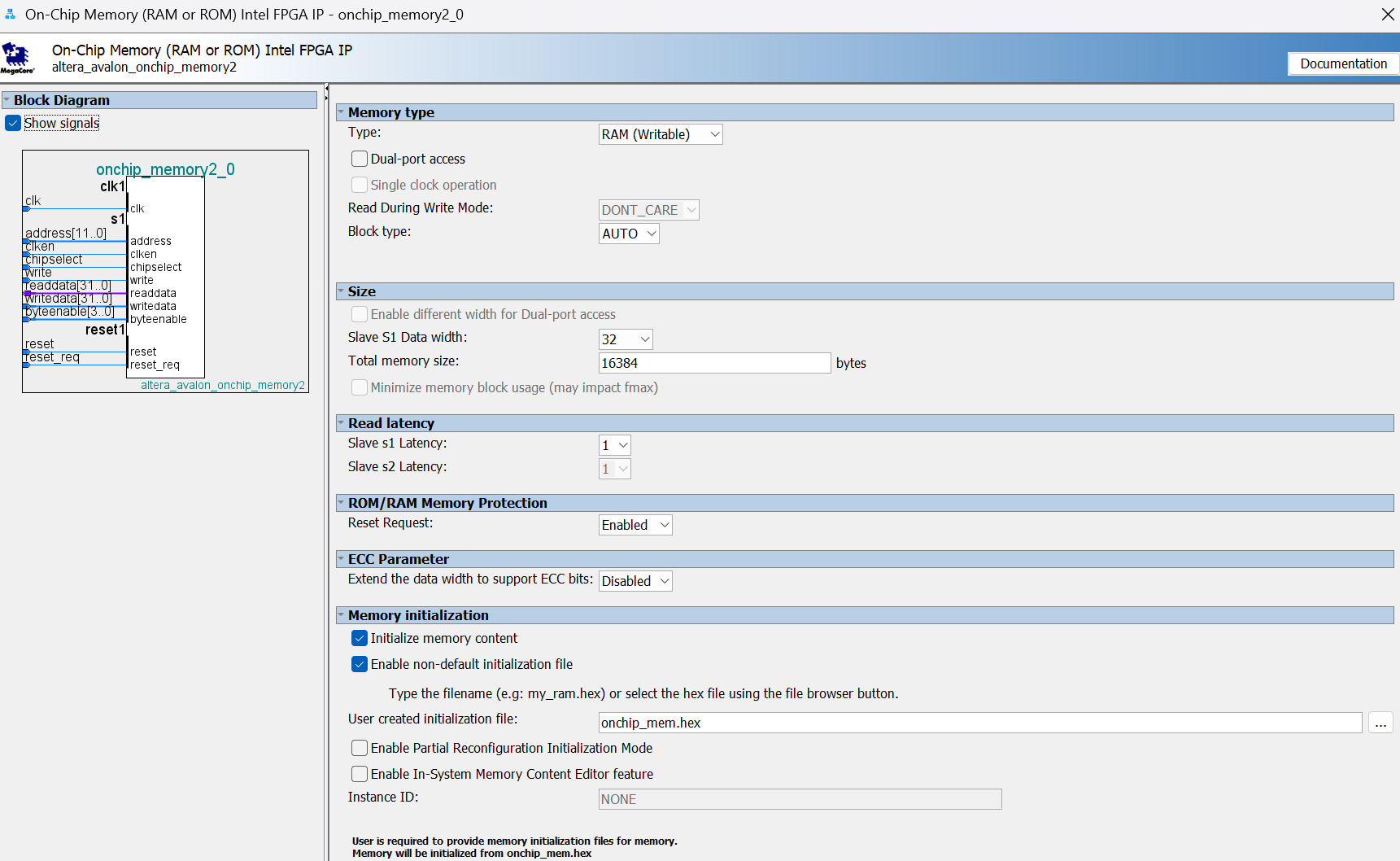


Рис. 5 – Настройка модуля onchip \_mem

Заметим, что появившиеся ошибки связаны с тем, что для нормальной работы компонент должен быть подключен к тактовому сигналу, сигналу сброса внутренних регистров и Мастеру на шине Avalon-ММ.

Подключим только что добавленный модуль. Для этого соединим выход clk.clk со входом clk1 компонента onchip\_mem, а также выход clk.clk\_reset со входом onchip\_mem.reset1:

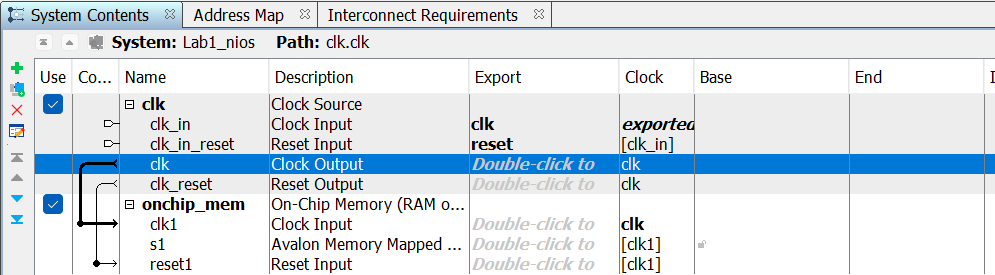


Рис. 6 – Подключение onchip\_mem

Как и предполагалось, ошибки исчезли.

### Настройка и подключение Nios II Processor

Добавим в систему ядро процессорного модуля, за него отвечает модуль Nios II Processor. Переименуем его на nios2\_PD и зададим в его настройках следующие параметры:

* Тип процессора Main → Тип процессора – NIOSII/e (простейший вариант процессорного ядра)
* JTAG DEBUG → отключить режим Include JTAG Debug

Изображение выглядит как текст, программное обеспечение, Шрифт, веб-страница

Автоматически созданное описание

Рис. 7 – Настройка модуля nios2\_PD

Подключим только что добавленный модуль. Для этого соединим вход nios2\_PD.clk с выходом clk.clk, а также выход clk.clk\_reset со входом nios2\_PD.reset. Соединим вход onchip\_mem.s1 с выходами data\_master и instruction\_master компонента nios2\_PD. Получим следующую систему:

Изображение выглядит как текст, программное обеспечение, число, линия

Автоматически созданное описание

Рис. 8 – Подключение модуля nios2\_PD

Дополнительно до настроим процессор NIOS II. Когда мы включаем или сбрасываем устройство, возникает некоторое событие Esception, которое говорит, что процессору нужно начать работать с определённого адреса. По этому адресу будет храниться первоначальная инициализация процессора. Поскольку, сброс, может быть нажат в любой момент, нужно указать следующие параметры:

* Память для вектора сброса
* Память для вектора exception
* JTAG Debug Module

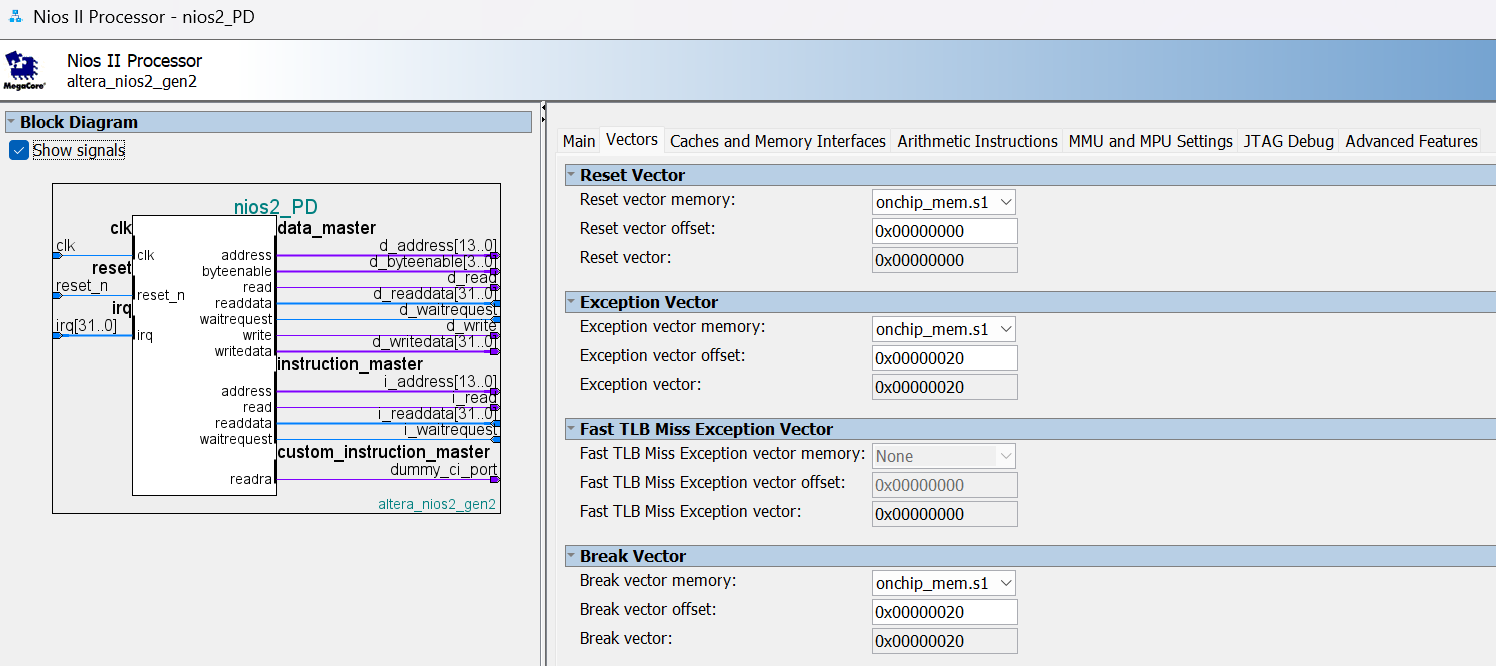


Рис. 9 – Настройка параметров exception в модуле nios2\_PD

### Настройка и подключение PIO (Parallel I/O)

#### pio\_LED

Через окно Library добавим в проект модуль PIO (Parallel I/O), переименуем его на pio\_LED и зададим следующие параметры:

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рис. 10 - Настройка модуля pio\_LED

Подключим только что добавленный модуль. Для этого выполним соединения так, как показано на рис. 12 ниже, а также в строке external\_connections столбца Export введём имя внешнего вывода – led:

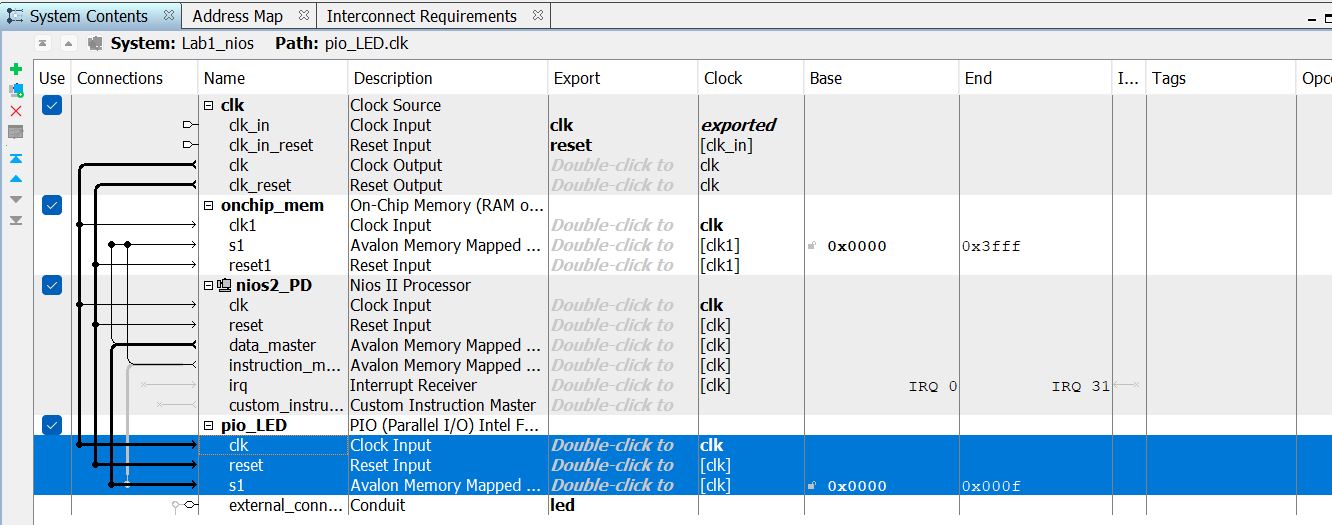


Рис. 11 – Подключение модуля pio\_LED

#### pio\_SW

Подключим второй модуль PIO (Parallel I/O), который будет отличаться от первого тем, что он будет иметь разрядность 1, а также направление передачи – Input. Переименуем его на pio\_SW. Настройки этого компонента приведены ниже на рис. 13:

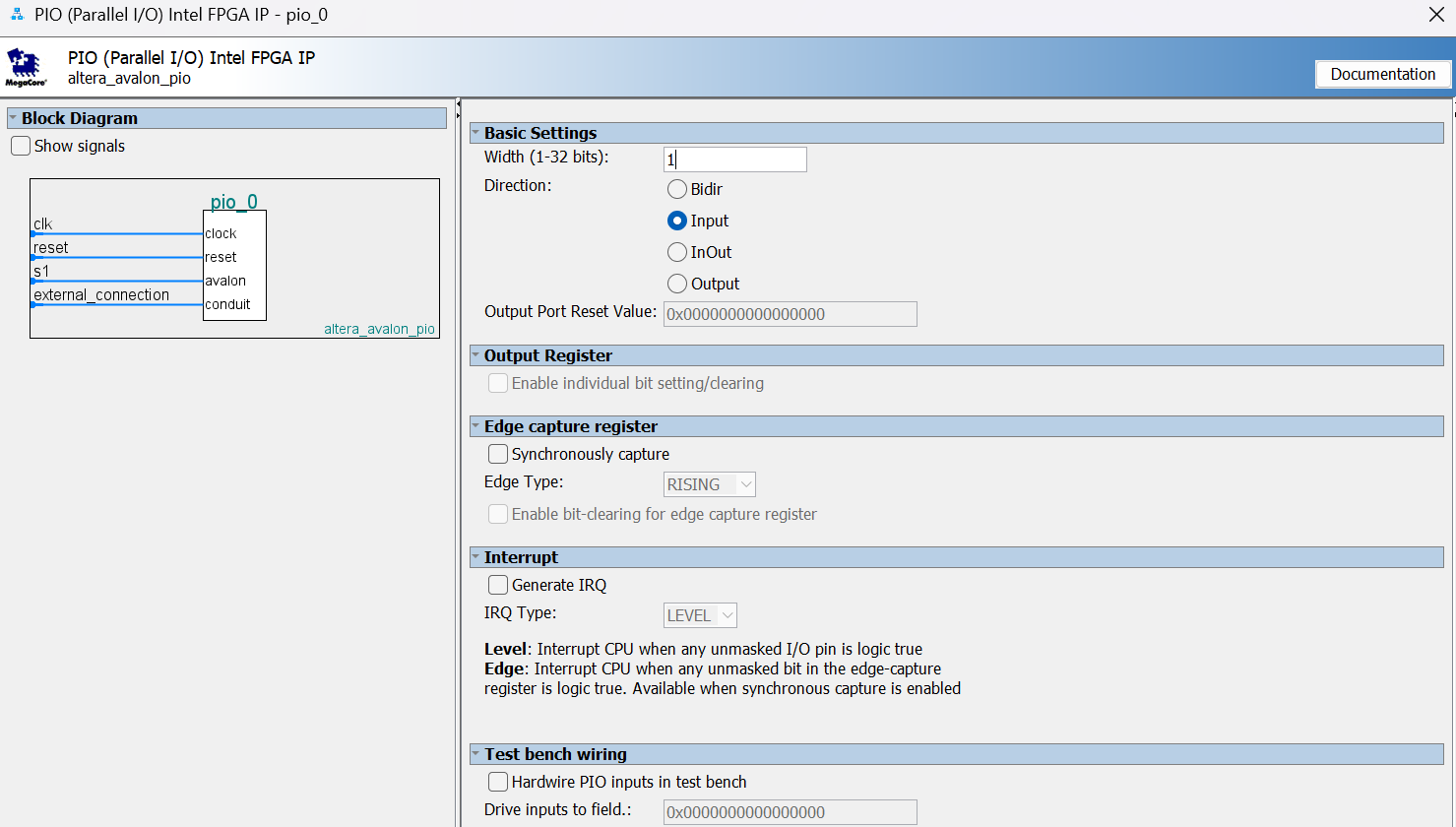


Рис. 12 – Настройка модуля pio\_SW

Подключим только что добавленный модуль. Для этого выполним соединения так, как показано на рис. 14 ниже, а также в строке external\_connections столбца Export введём имя внешнего вывода – sw:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рис. 13 – Подключение модуля pio\_SW

### Адресация системы

Выполним System → Assign base Addresses. После автоматической адресации модулей окно Address Map будет выглядеть следующим образом:

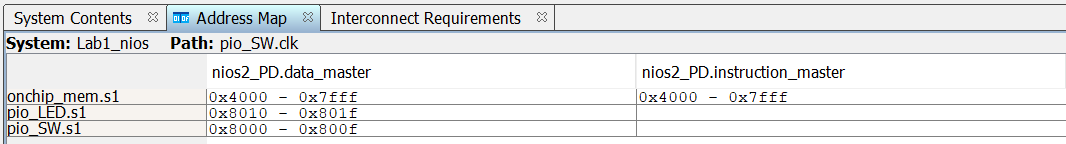


Рис. 14 – Окно Address Map после выполнения автоматической адресации

## Анализ полученной системы

Полученная система будет выглядеть так, как показано на ниже:

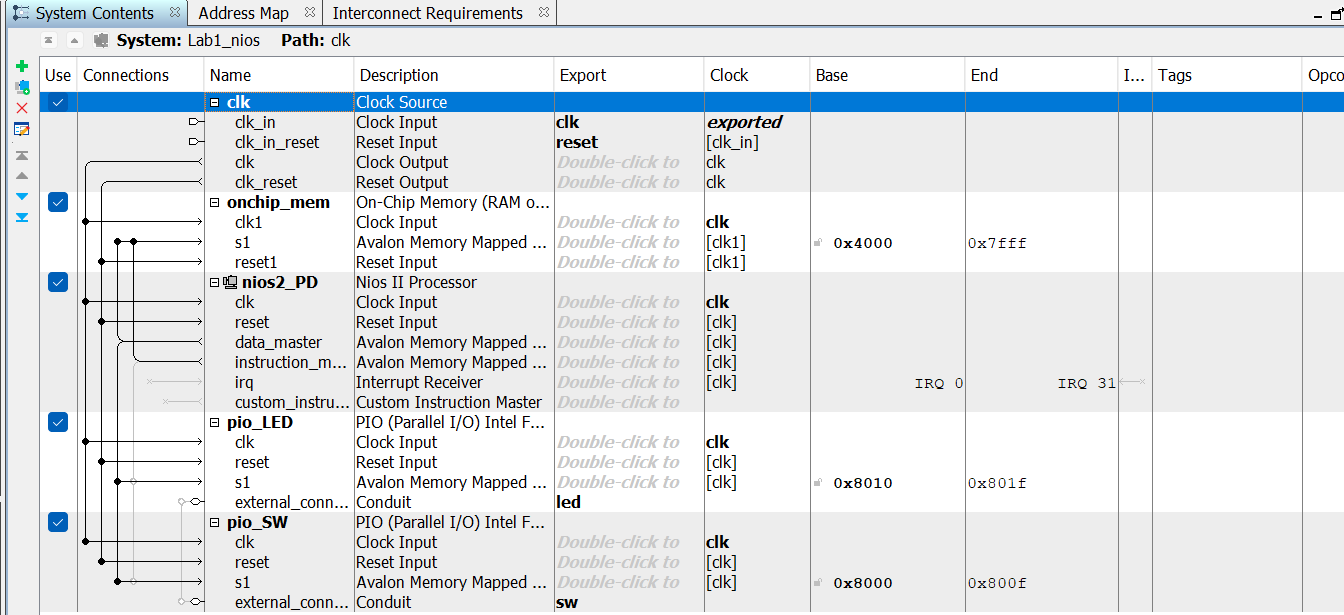


Рис. 15 – Внешний вид созданной системы

### Анализ предустановок

Проверим, что предустановки для полученной системы указаны так, как показано на рис. 17 ниже:

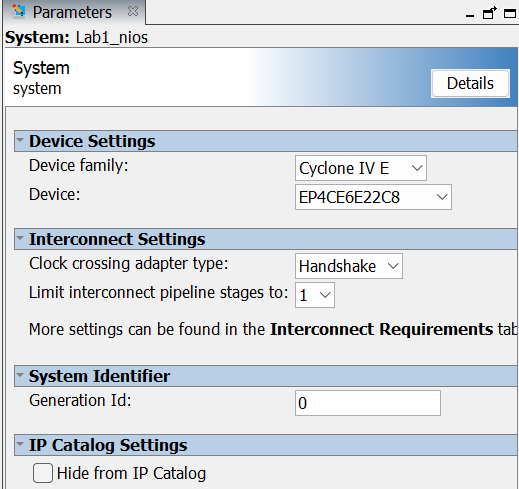


Рис. 16 – Предустановки системы

В окне Messages есть только 1 предупреждение, связанное с тем, что не подключён JTAG Debug модуль, однако это было сделано намеренно, поэтому на это предупреждение можем не обращать внимание:

Изображение выглядит как текст, линия, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 17 – Окно Messages с предупреждением

### Проверка блока

Выполним View → Block Symbol и убедимся в том, что символ системы построен правильно:

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рис. 18 – Символ системы

### Проверка отсутствия проблемных подключений

Выполним View → Clock domains Beta, выберем режим отображения Reset, затем Clock. Заметим, что проблемных подключений ни в одном из случаев не выявлено:

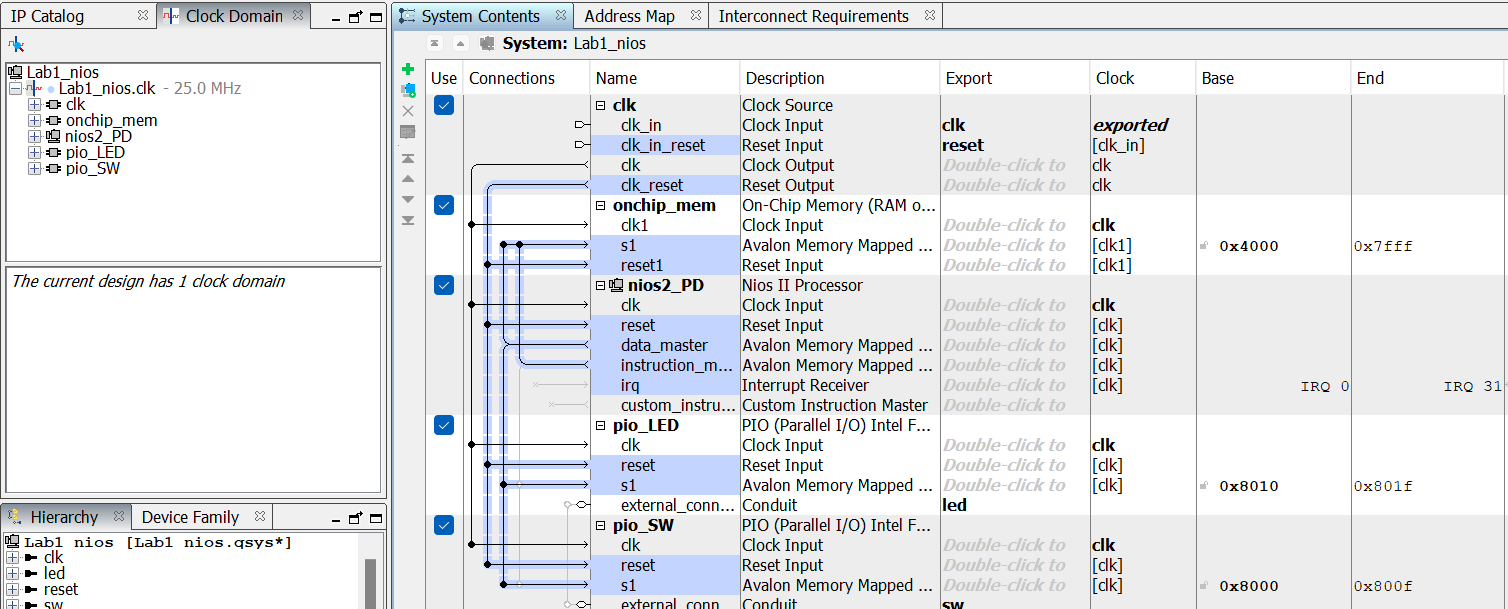


Рис. 19 – Анализ проблемных подключений

Выполним команду System → Show System with PD Interconnect (Show System with QSYS Interconnect). Проверим, был добавлен только модуль mm\_interconnect\_0.

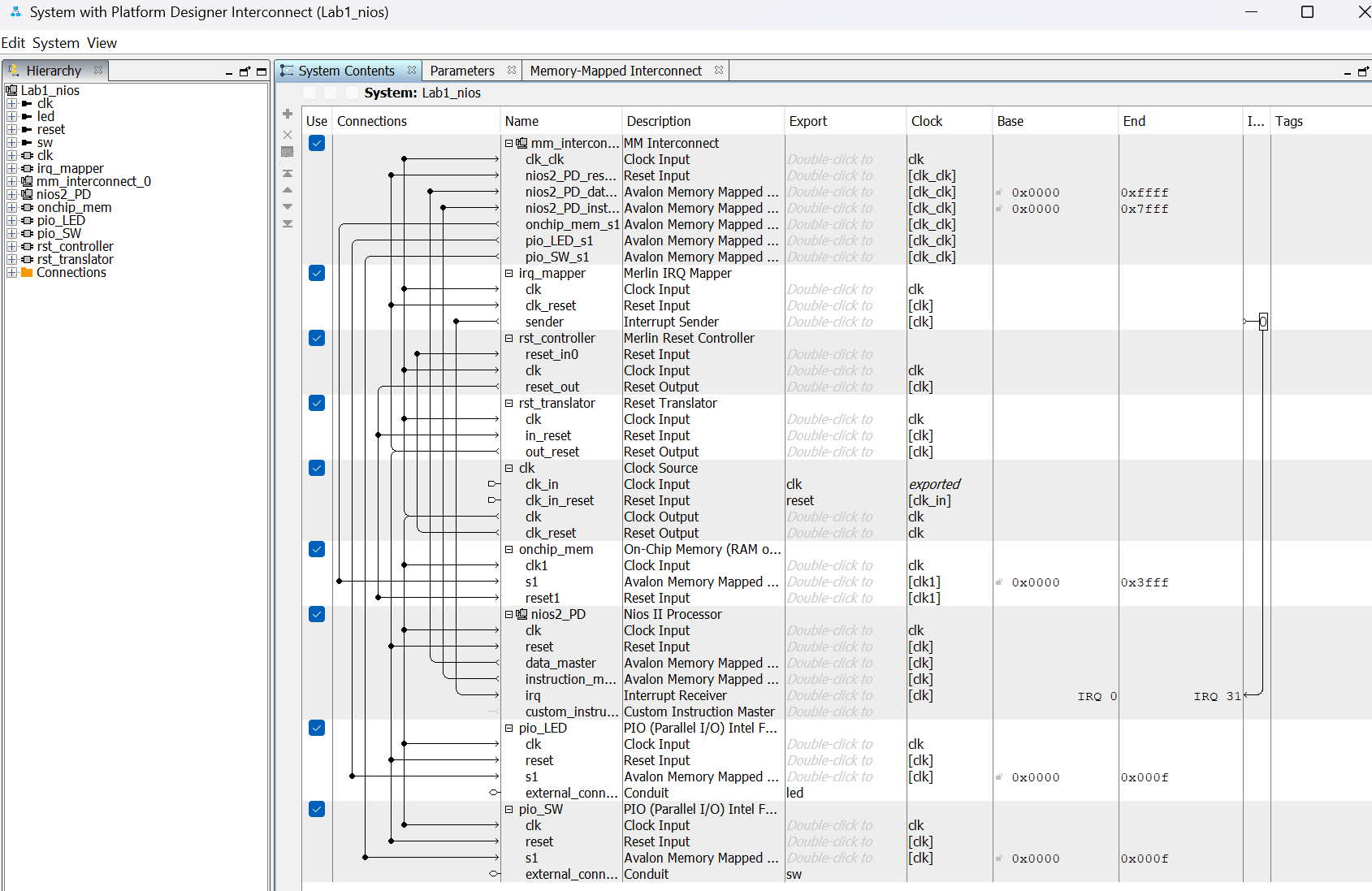


Рис. 20 – Show System with QSYS Interconnect

### Анализ с помощью Schematic

Выполним View → Schematic, в качестве фильтра введём in и убедимся в том, что система синхронизации и каналы ST системы подключены верно:

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис. 21 – Schematic

### Генерация системы

Выполним PD → Generate HDL и укажем следующие предустановки для генерации:

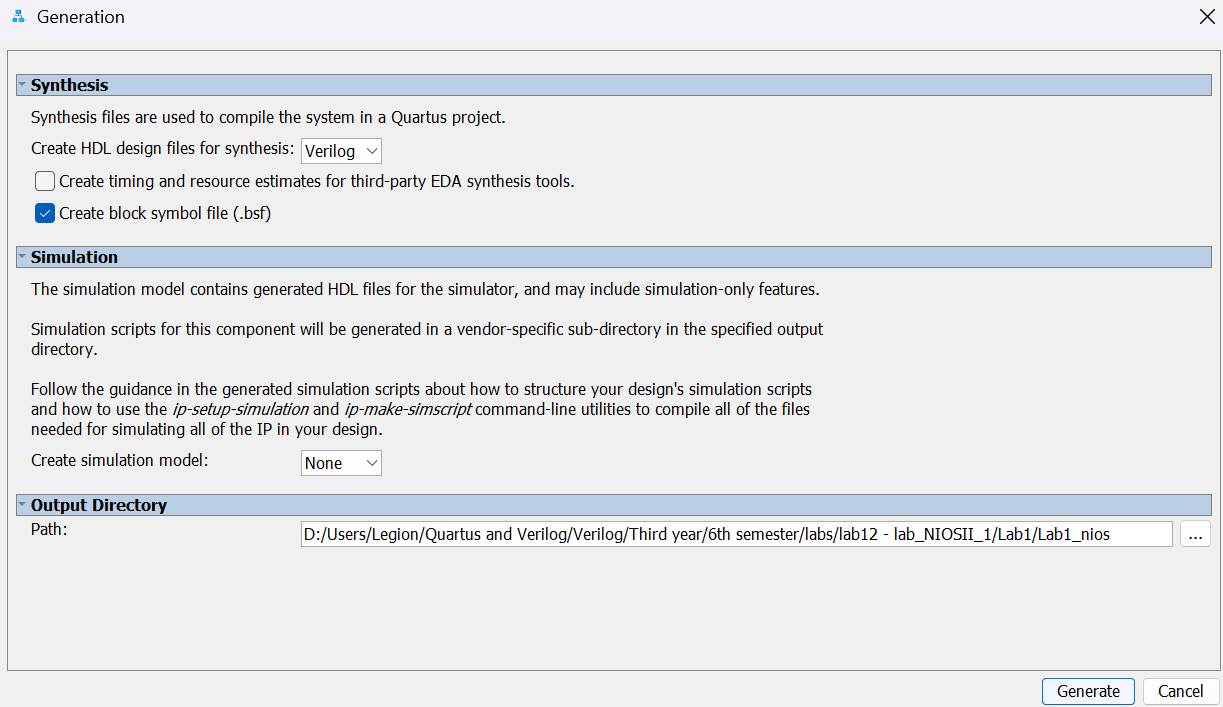


Рис. 22 – Предустановки окна Genreration (по умолчанию)

Удостоверимся в том, что среди предупреждений есть только 1 пункт, связанный с JTAG, про сто говорилось выше:

Изображение выглядит как текст, Шрифт, программное обеспечение, веб-страница

Автоматически созданное описание

Рис. 23 – Проверка генерации HDL

## Подключение файлов к проекту

Подключим файл .qip только что созданной системы к проекту в Quartus

Изображение выглядит как текст, снимок экрана, программное обеспечение, линия

Автоматически созданное описание

Рис. 24 – Подключение файла .qip к проекту

Создадим файл верхнего уровня Lab1.sv, синтаксис которого приведён ниже:

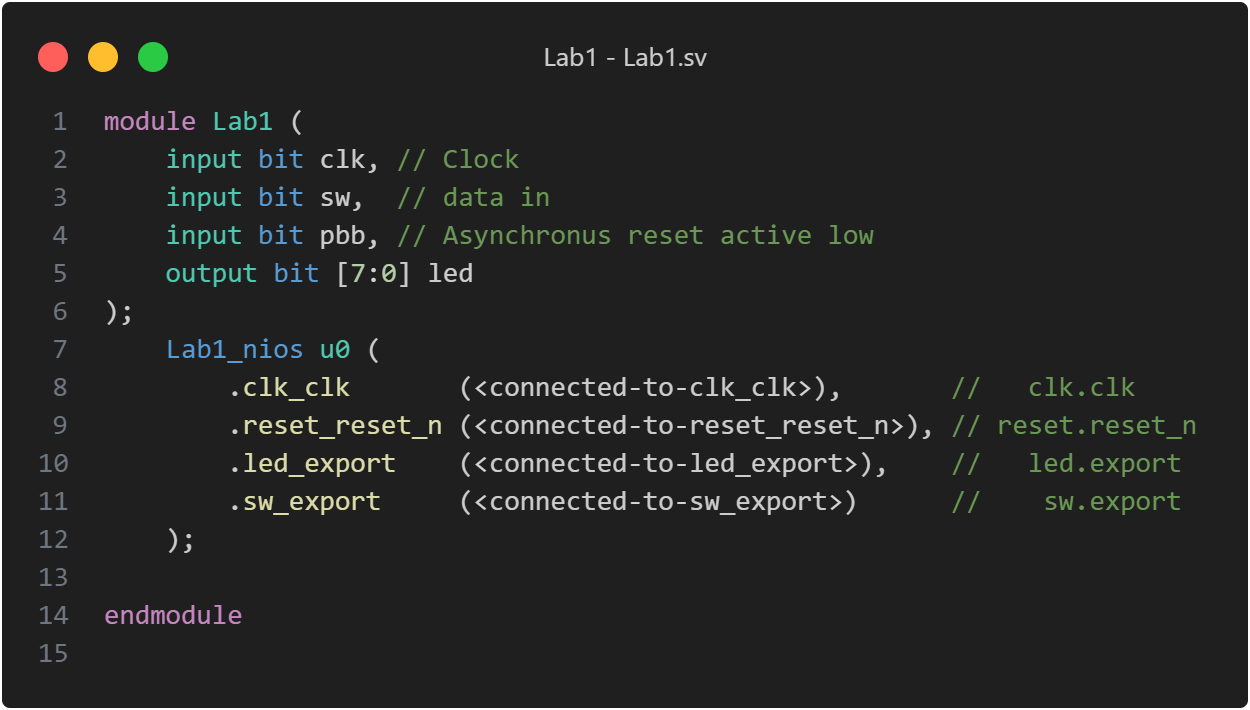


Рис. 25 – Синтаксис файла Lab1.sv

Этот файл был создан отдельно, но не добавлен в проект. Создадим этот же файл, но с помощью графического редактора:



Рис. 26 – Создание символа файла верхнего уровня

Сохраним схему на Рис. 27 под именем lab1.bdf

## Анализ проекта

В иерархии проекта выберем файл верхнего уровня:

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Автоматически созданное описание

Рис. 27 – Выбор файла верхнего уровня

### RTL Viewer

Выполним анализ и синтез проекта средствами QP и убедимся в правильности схемы средствами RTL Viewer:

Изображение выглядит как текст, диаграмма, линия, План

Автоматически созданное описание

Рис. 28 – Схема проекта в RTL Viewer

Можем увидеть, что полученная в RTL Viewer схема совпадает с той, что была задана по условию (в зелёном блоке отображается тот фрагмент системы, который был создан средствами PD).

### Назначение выходов проекта

Откроем редактор назначения выводов (Pin Planner): Assignment → Pin Planner. Назначим выводы на плату так, как показано на ниже:

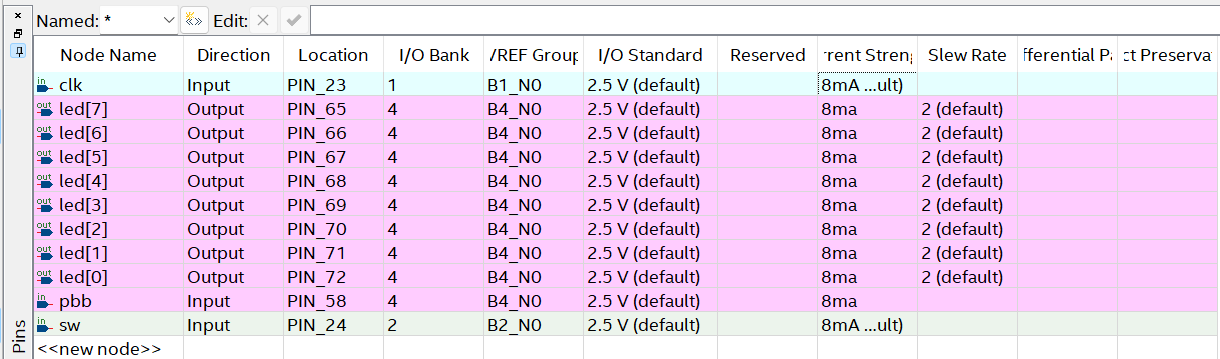


Рис. 29 – Назначение выводов платы средствами Pin Planner

Выполним Assignment → Device → Device and Pin Options → Unused Pins, в появившемся окне установим опцию As input tri-started with weak pull-up:

Изображение выглядит как текст, программное обеспечение, число, Значок на компьютере

Автоматически созданное описание

Рис. 30 – Unused Pins

## Создание программной части проекта

Создадим оболочку Nios II SBT (Software Build Tools) средствами Eclipse.

Укажем файл с описанием программы lab1\_nios.spocinfo. В качестве названия проекта укажем Lab1\_sw.

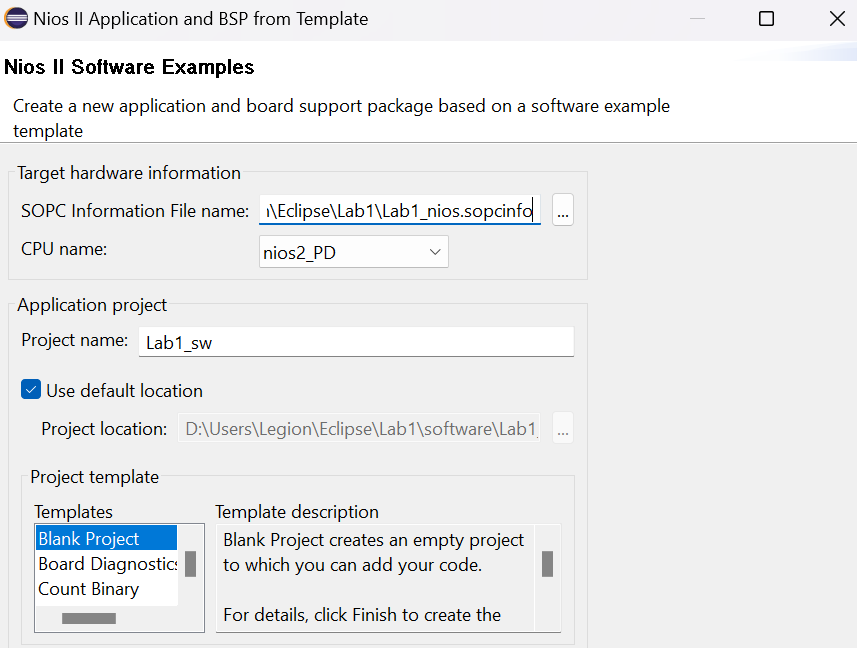


Рис. 31 – Создание проекта в Eclipse

### Создание файла Source

Создадим Source File:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 32 – Пред настройки Source File’а

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 33 – Синтаксис файла Lab1\_source.c

### Компиляция проекта в Eclipse

Выполним Lab1\_sw → Build Project:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 34 – Build Project

При создании платформы, программой и данными инициализации занято 4568 Байт, свободно 10 кБ.

Выполним Lab1\_sw\_bsp → Nios II Lab1 → BSP Editor и в окне настроек зададим параметры следующим образом:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 35 – Настройка BSP Editor

Скомпилируем проект. Для этого выполним Lab1\_sw → Build Project, проверим, что останется больше свободной памяти:

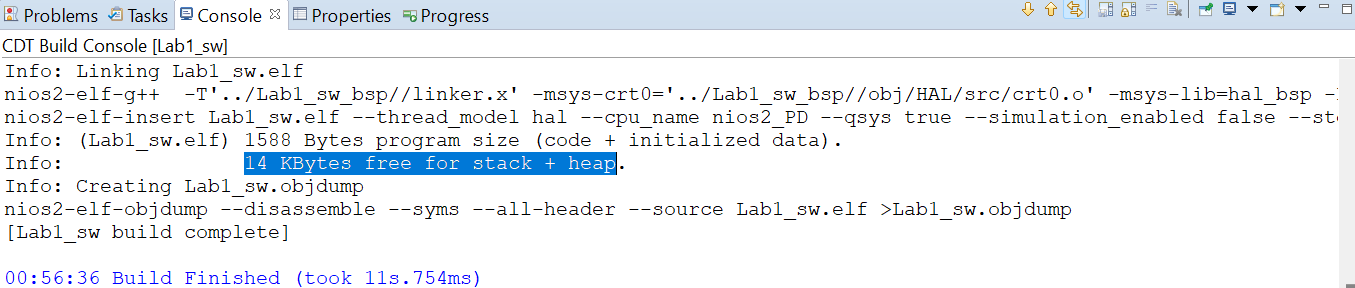


Рис. 36 – Build Project после изменения настроек BSP Editor

Заметим, что памяти, оставшейся свободной (из ОЗУ 16 Кбайт, указанных при создании платформы, программой и данными инициализации занято 1548 Байт (было - 4568 Байт), свободно 14кБайт ( было - 10кБайт).

Выполним Lab1\_sw → Make Targets → Build:

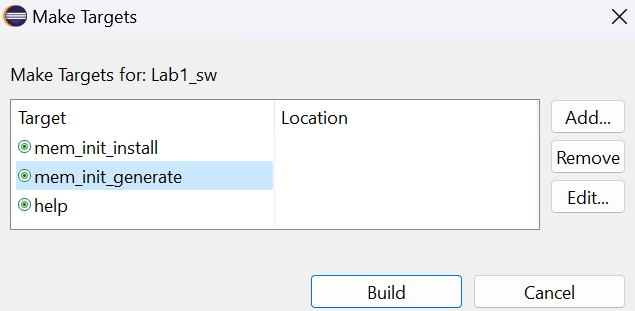


Рис. 37 – Окно Make Targets

Компиляция прошла успешно, проверим, что размер и базовый адрес памяти совпадают с теми, что были указаны ранее при создании системы в пакете PD:

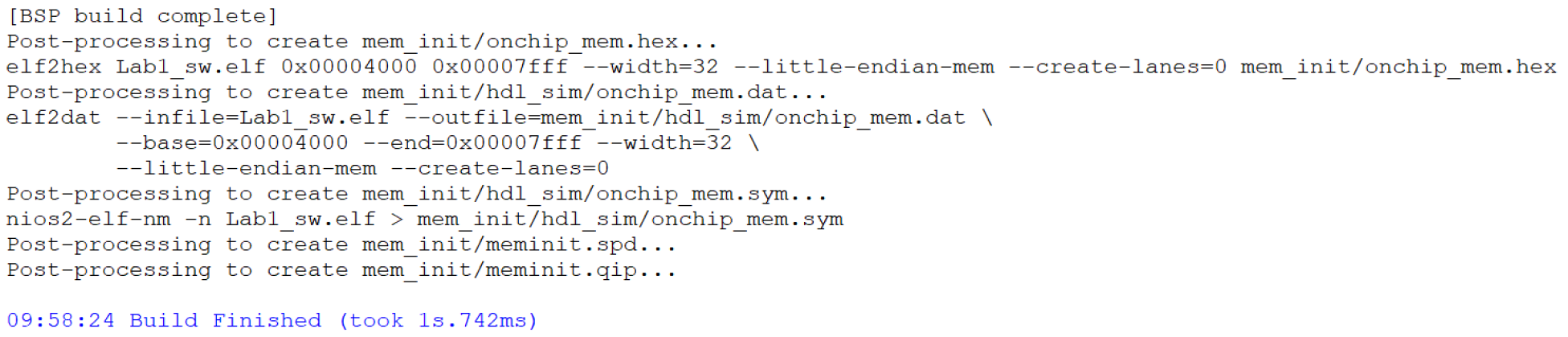


Рис. 38 – Проверка размера и базового адреса памяти

Как видим, все значения совпали с теми, что были указаны ранее.

### Полная компиляция проекта в QP

Добавим только что созданный файл meminit.qip в проект:

Изображение выглядит как текст, Шрифт, число, программное обеспечение

Автоматически созданное описание

Рис. 39 – Добавление meminit.qip в проект

Создадим Lab1.sdc файл с временными требованиями так, как показано на Рис. 40 ниже:

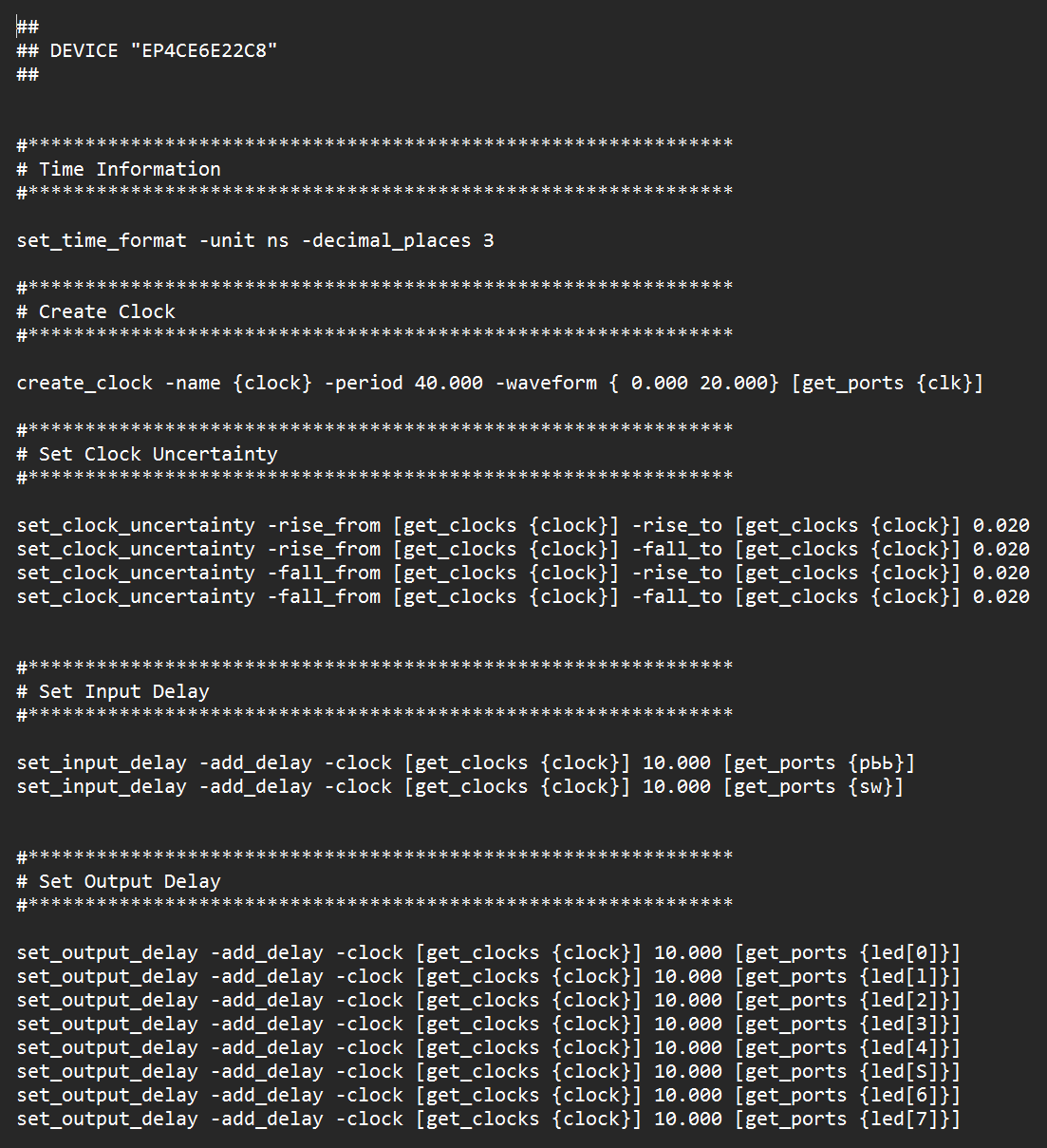


Рис. 40 – SDC файл

Добавим файл к проекту:

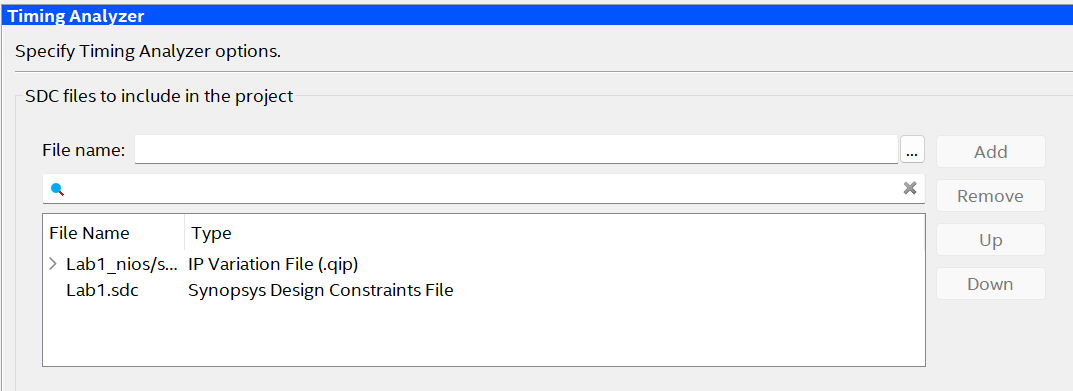


Рис. 41 – Добавление SDC файла

Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

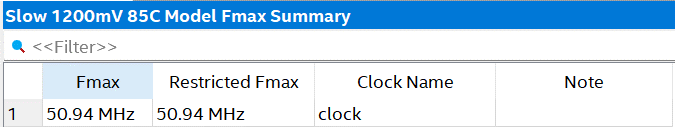


Рис. 42 – Временные характеристики устройства

## Реализация на плате

Проект был загружен на плату, светодиоды LED8-LED1 отображали последовательное увеличение счётчика от 0 до 255 при sw[0] в положении 1, а при переключении sw[0] в положение 0 счёт происходил в обратном порядке с тем же шагом от 255 до 0. Работа на стенде была продемонстрирована преподавателю.

# Дополнительные задания

## Дополнительное задание 1

### Создание программного файла Lab1\_task1\_source.c

Создадим отдельный проект аналогичный предыдущему, но изменим в нём файл с исходным кодом Lab1\_task1\_source.c. Будем использовать указатели в качестве адресов, а также увеличим частоту переключения светодиодов засчёт уменьшения значения функции usleep, изменим начальное значение счётчика

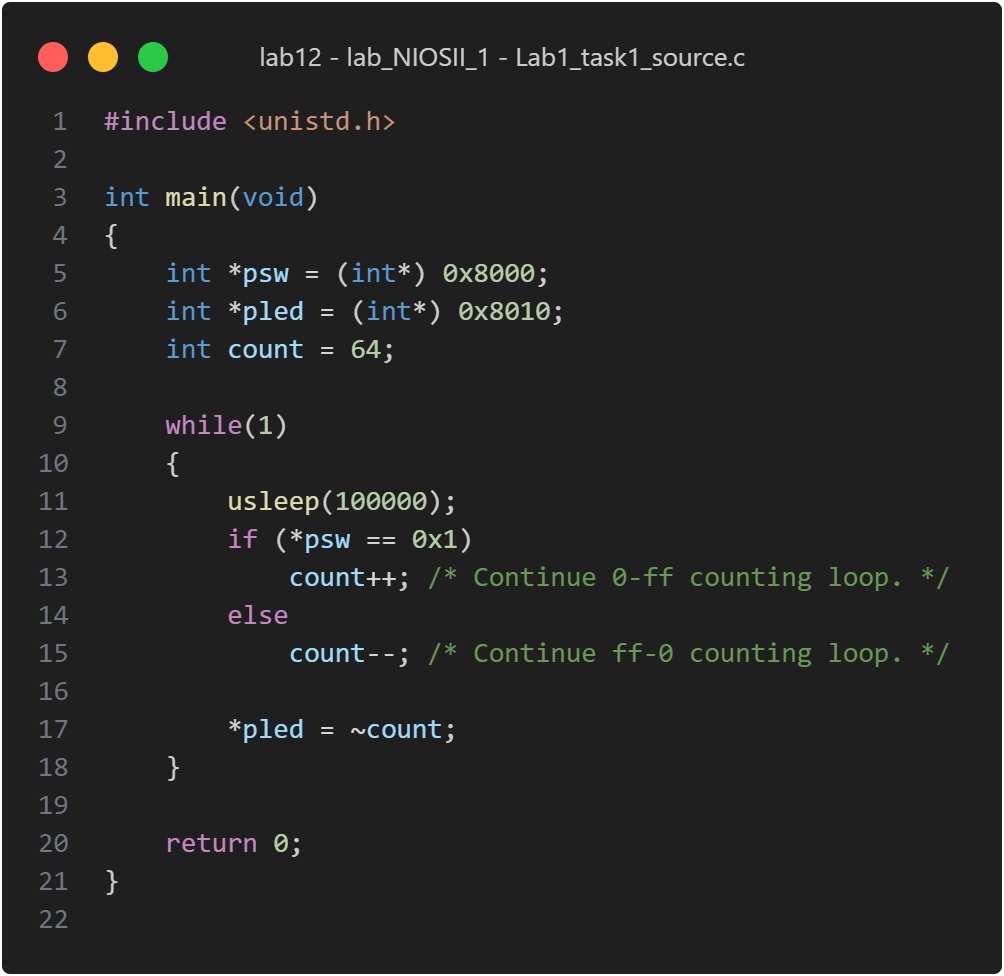


Рис. 43 – Изменённый исходный файл

### Реализация на плате

Как и ожидалось, визуально можем наблюдать корректную работу счётчика и правильное отображение на лабораторном стенде, аналогично тому, что было описано в основном задании проекта. Визуально заметно, что переключение светодиодов, засчёт увеличения частоты, происходит быстрее.

## Дополнительное задание 2

Создадим второй проект, где в системе изменим разрядность SW с одного разряда до 8 разрядов, подключим к переключателям SW[7:0].

### Увеличение разрядности pio\_SW

Изменим разрядность модуля pio\_SW с 1 на 8:

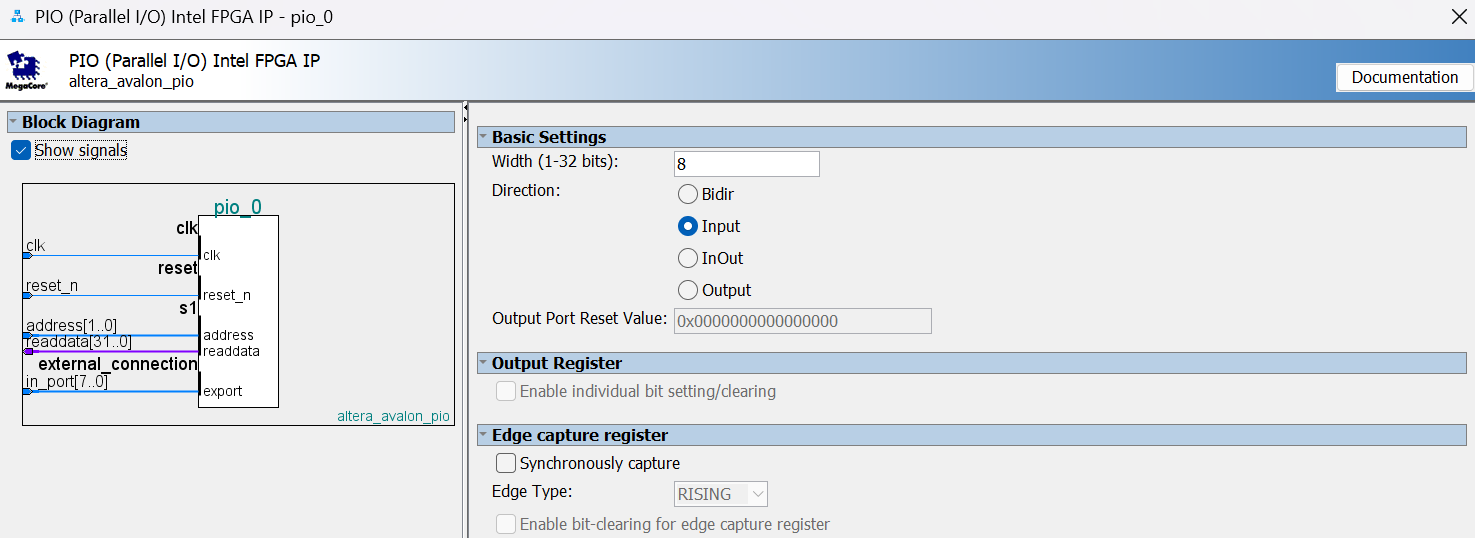


Рис. 44 – Изменение настроек модуля pio\_SW

Подключим новые sw к переключателям SW[7:0] средствами Pin Planner:

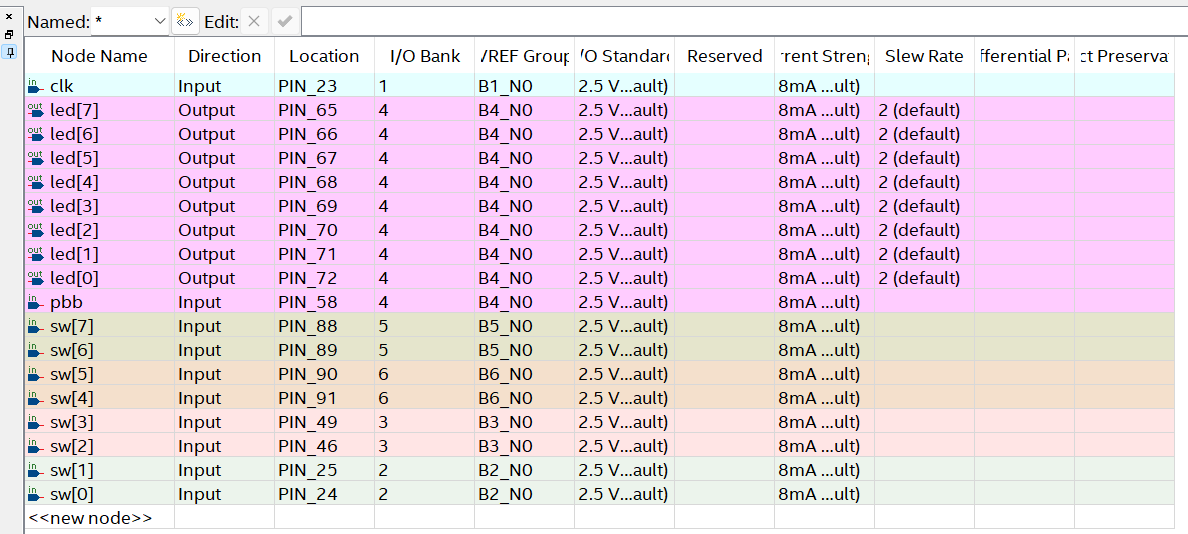


Рис. 45 – Подключение SW[7:0] (Pin Planner)

### Создание программного файла Lab1\_task2\_source.c

Создадим текст программы в файле с исходным кодом Lab1\_task2\_source.c, где будет обеспечиваться сложение по модулю, задаваемому переключателями SW[7:0]. Синтаксис файла представлен на Рис. 47 ниже:

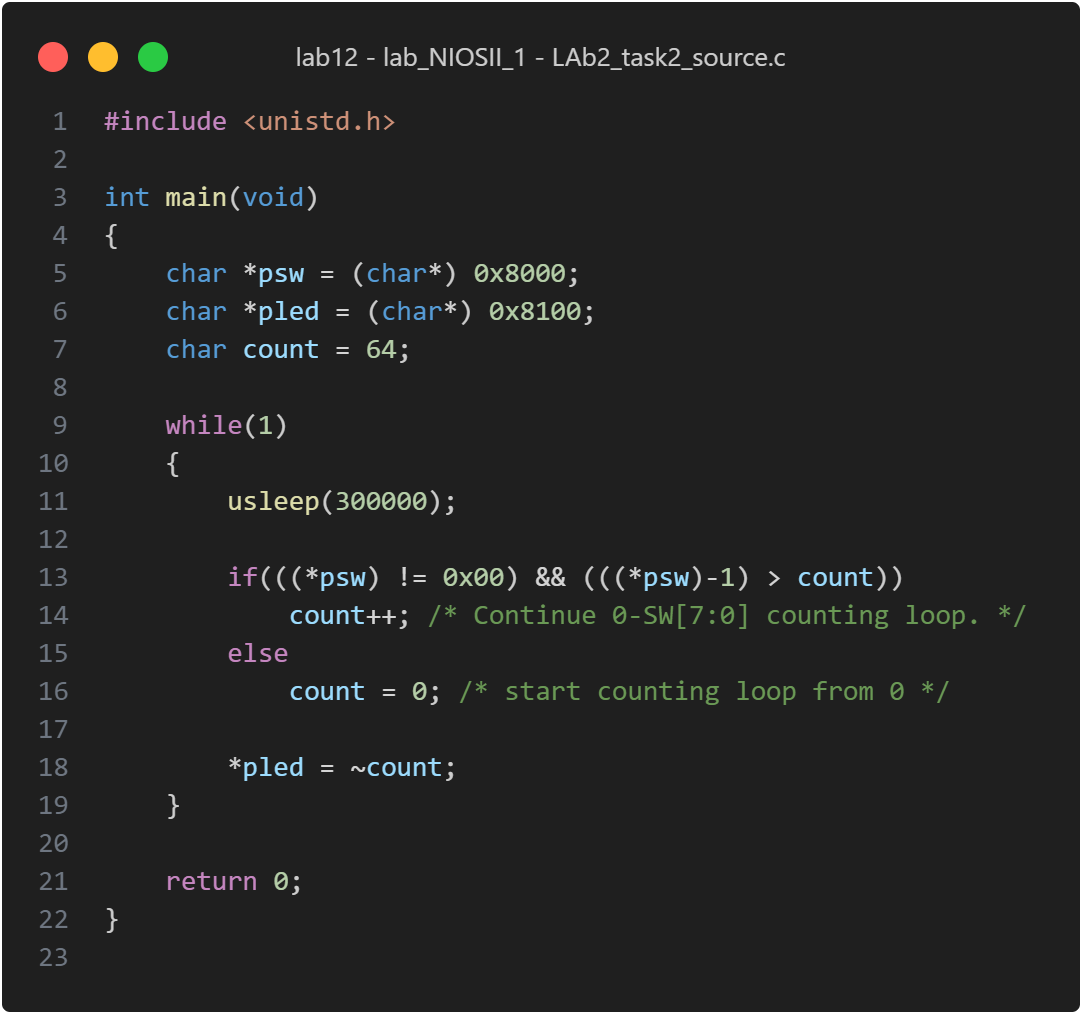


Рис. 46 – Программный файл с реализацией сложения по модулю SW[7:0]

### Реализация на плате

Проект был загружен на плату. Увеличение счётчика по модулю работает корректно.

Так, например, при выставлении значения на SW[7..0] = 12 осуществляется увеличение счётчика от 0 до 11. Это связано с тем, что → выполняется условие на 16 строке (значение счётчика по умолчанию меняется с 64 на 0).

Если же изначально установить значение на SW[7:0] = 70, то счёт будет производиться от 64 до 69.

Работа с устройством на плате была продемонстрирована преподавателю.

# Вывод

В ходе лабораторной работы был осуществлен процесс создания и настройки системы (Рис. 1) на базе процессора NIOS II с использованием пакета Quartus Prime и Eclipse IDE. Реализация проекта включала в себя создание аппаратной части с помощью Platform Designer (PD), создание программной части с помощью среды разработки Eclipse, где был создан файл для инициализации модуля памяти программ процессора, а также реализация проекта на плате.

В ходе тестирования проекта на плате светодиоды LED8-LED1 отображали последовательное увеличение счётчика от 0 до 255 при sw[0] в положении 1, а при переключении sw[0] в положение 0 счёт происходил в обратном порядке с тем же шагом от 255 до 0. Это говорит о корректности работы созданного устройства.

В дополнительных заданиях были реализованы различные функциональности, такие как изменение частоты переключения светодиодов, использование указателей для адресации данных, а также увеличение разрядности ввода-вывода для взаимодействия с переключателями.

Анализ результатов показал успешную работу созданной системы на плате, а также корректное отображение данных на светодиодах в зависимости от значений на переключателях.

Полученные навыки работы с NIOS II могут быть полезны при создании проектов в области встраиваемых систем, таких как системы управления, обработки сигналов, автоматизации и других.