Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт курсовой работе. Часть 2**

Дисциплина: Автоматизация проектирования дискретных  
устройств (на английском языке).

Выполнил студент гр. 5130901/10101 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Л. Симоновский (подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Федотов (подпись)

“08” мая 2024 г.

Санкт-Петербург

2024

Оглавление

[1. Список иллюстраций: 2](#_Toc165982703)

[2. Проект 1: 3](#_Toc165982704)

[2.1. Структура проекта: 3](#_Toc165982705)

[2.2. Решение: 3](#_Toc165982706)

[3. Проект 2: 15](#_Toc165982707)

[3.1. Структура проекта: 15](#_Toc165982708)

[3.2. Решение: 15](#_Toc165982709)

[4. Вывод: 15](#_Toc165982710)

# Список иллюстраций:

[Рис. 2.1. Структура разрабатываемого проекта. 3](#_Toc165980334)

[Рис. 2.2. Настройки для QP. 3](#_Toc165980335)

[Рис. 2.3. Настройки модуля счетчика. 4](#_Toc165980336)

[Рис. 2.4. Настройка модуля памяти. 4](#_Toc165980337)

[Рис. 2.5. Модуль ввода для светодиодов. 5](#_Toc165980338)

[Рис. 2.6. Модуль ввода для кнопки. 5](#_Toc165980339)

[Рис. 2.7. Модуль JTAG UART. 6](#_Toc165980340)

[Рис. 2.8. Настройки вектора для Nios II. 6](#_Toc165980341)

[Рис. 2.9. Platform Designer. 6](#_Toc165980342)

[Рис. 2.10. Schematic в Platform Designer. 7](#_Toc165980343)

[Рис. 2.11. System Interconnections. 7](#_Toc165980344)

[Рис. 2.12. RTL Viewer. 8](#_Toc165980345)

[Рис. 2.13. Pin Planner. 8](#_Toc165980346)

[Рис. 2.14. Результат компиляции. 9](#_Toc165980347)

[Рис. 2.15. Создание проекта для Nios II. 10](#_Toc165980348)

[Рис. 2.16. Создание файла с исходным кодом. 10](#_Toc165980349)

[Рис. 2.17. Настройки BSP. 11](#_Toc165980350)

[Рис. 2.18. Компиляция программы. 11](#_Toc165980351)

[Рис. 2.19. Приветственные сообщения после запуска. 11](#_Toc165980352)

[Рис. 2.20. Консоль после нескольких нажатий на pba. 12](#_Toc165980353)

[Рис. 2.21. Новые настройки для BSP. 12](#_Toc165980354)

[Рис. 2.22. Компиляция проекта с новыми настройками BSP. 12](#_Toc165980355)

[Рис. 2.23. Результат компиляции. 13](#_Toc165980356)

[Рис. 2.24. Optimization level: size. У lab3\_sw\_bsp. 13](#_Toc165980357)

[Рис. 2.25. Optimization level: size. У lab3\_sw. 14](#_Toc165980358)

[Рис. 2.26. Компиляция после выставления настроек оптимизации. 14](#_Toc165980359)

[Рис. 2.27. Результат запуска программы и нескольких переключений pba. 14](#_Toc165980360)

# Проект 1:

## Структура проекта:

Структура разрабатываемого проекта приведена ниже:

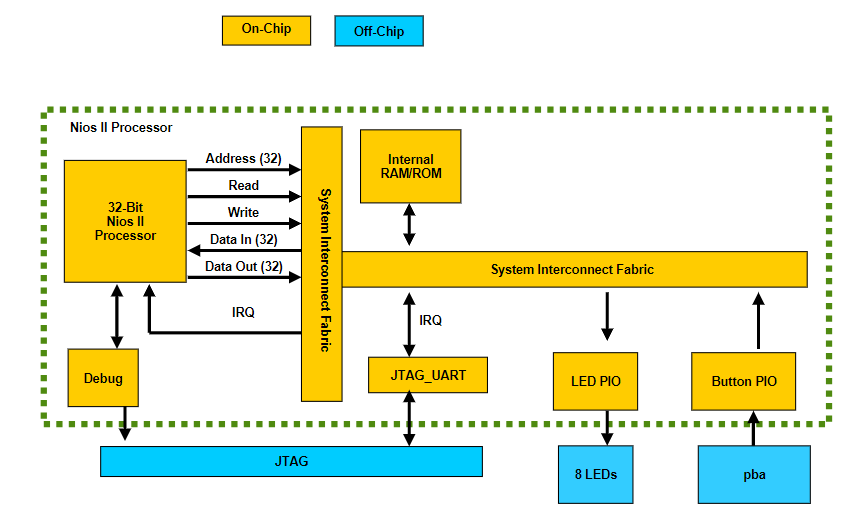


Рис. 2.1. Структура разрабатываемого проекта.

Под управлением процессора NIOSII обеспечивается:

* Опрос состояния кнопки pba.
* Формирование на консоли сообщений о нажатой кнопке.
* При каждом нажатии кнопки pba происходит изменение номера включенного светодиода от led1 к led8 на одну позицию (с циклическим переходом от led8 к led1).

## Решение:

Выполним создание проекта в QP со следующими настройками:

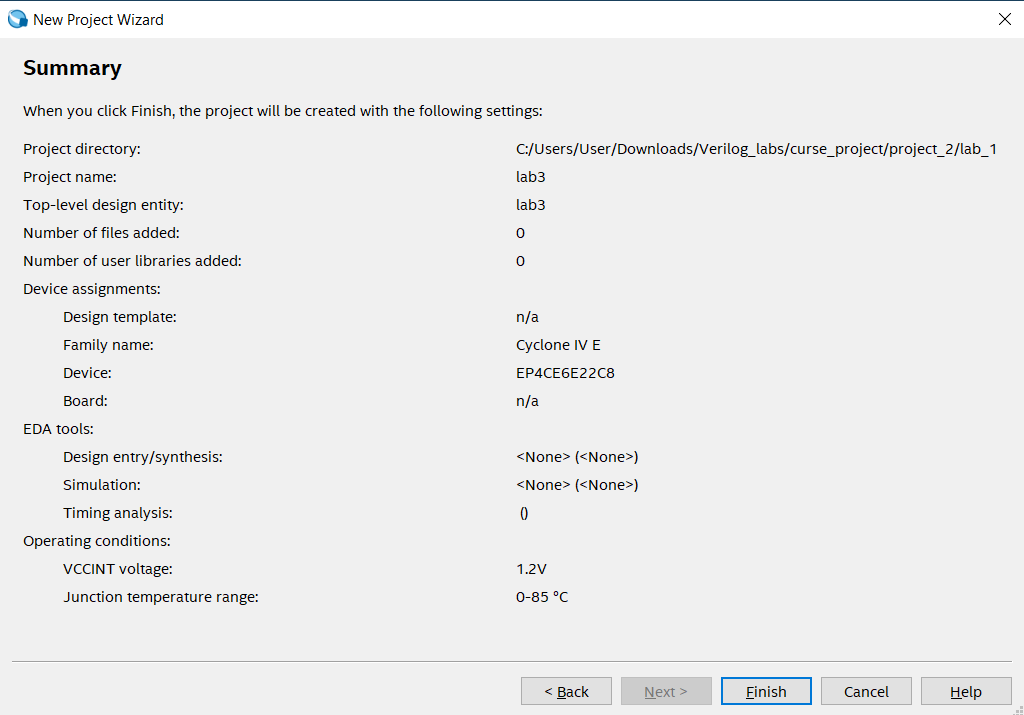


Рис. 2.2. Настройки для QP.

Далее переходим в Platform Designer и начинаем реализовывать схему с Рис. 2.1.

Начнем с модуля clk:

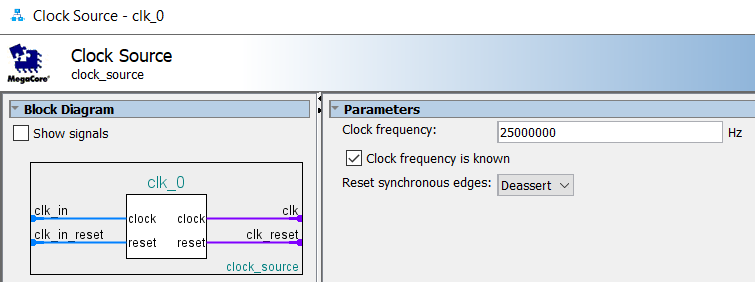


Рис. 2.3. Настройки модуля счетчика.

Добавим модуль памяти для процессора Nios II со следующими настройками:

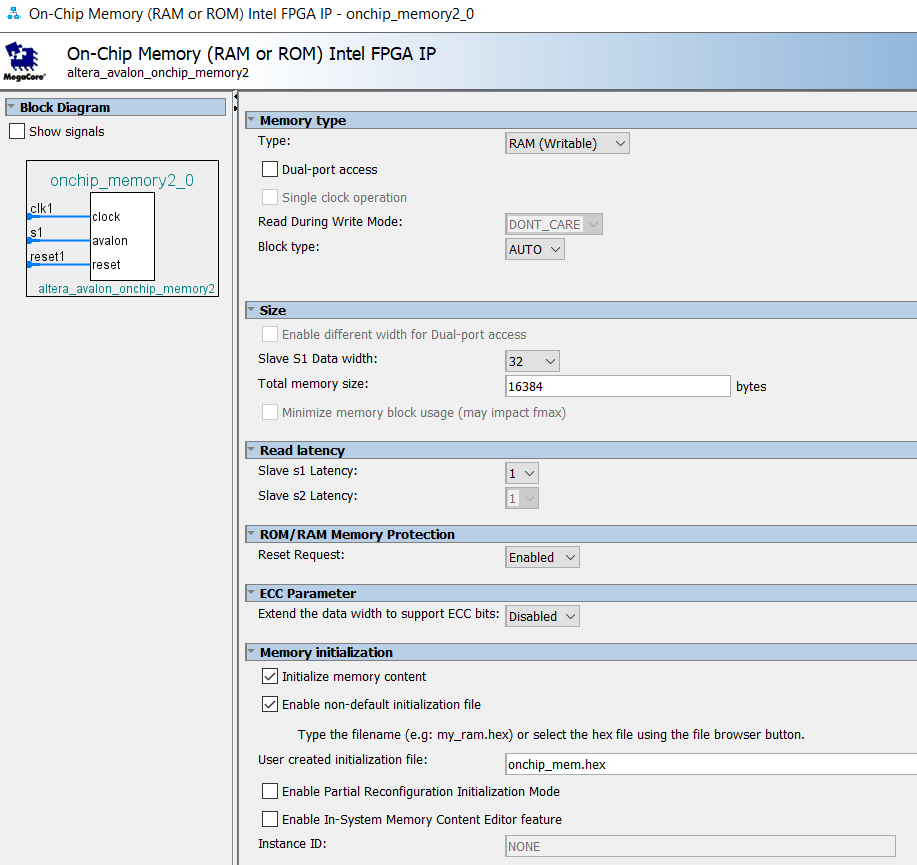


Рис. 2.4. Настройка модуля памяти.

Также необходим модуль вывода для светодиодов, добавим его со следующими настройками:

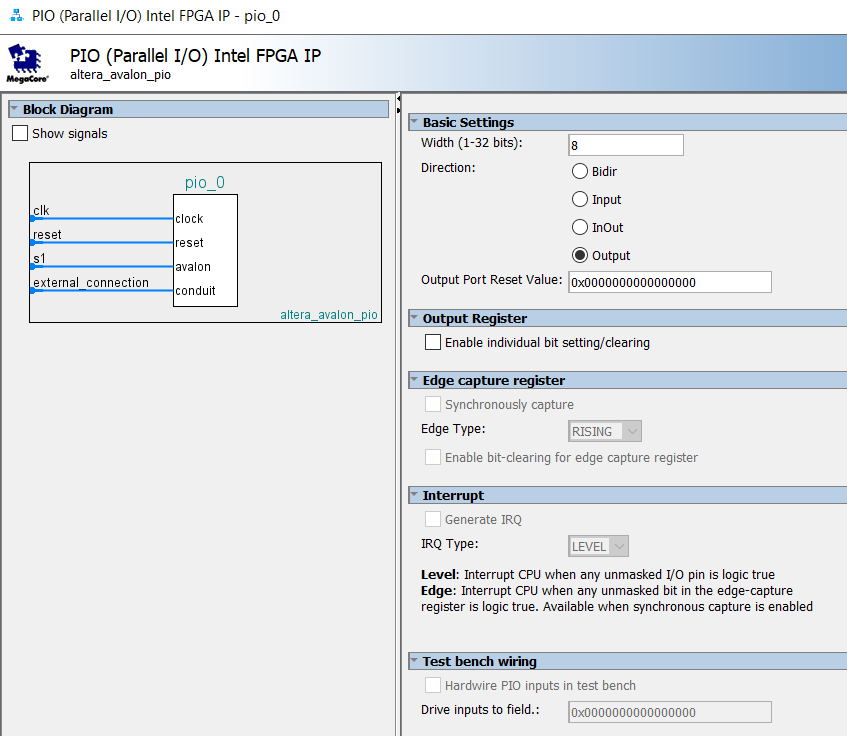


Рис. 2.5. Модуль ввода для светодиодов.

И модуль ввода для кнопки:

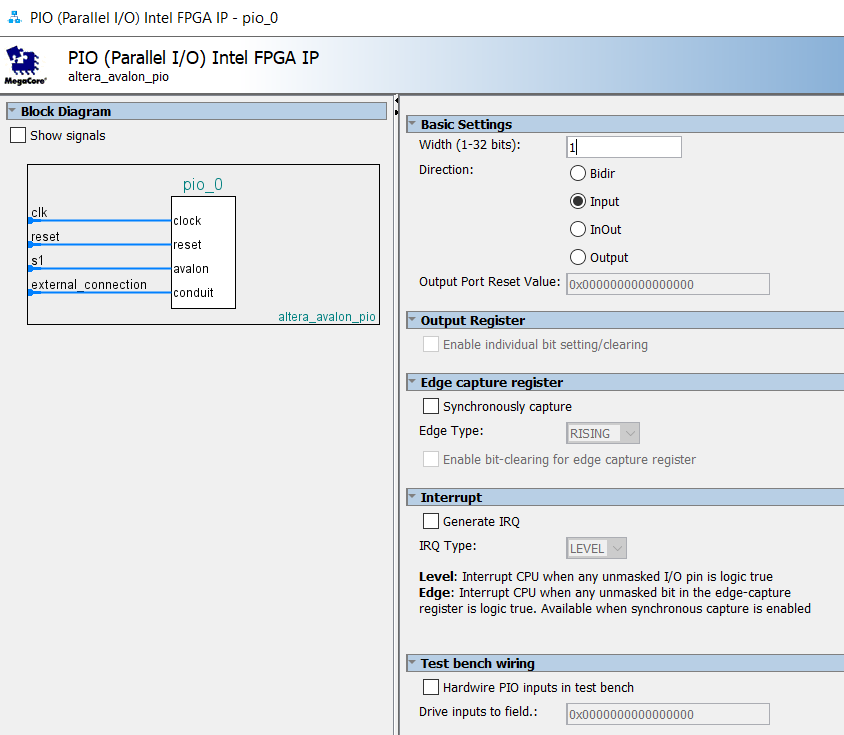


Рис. 2.6. Модуль ввода для кнопки.

Для того, чтоб процессор мог отправлять нам какие-то данные, добавим модуль UART:

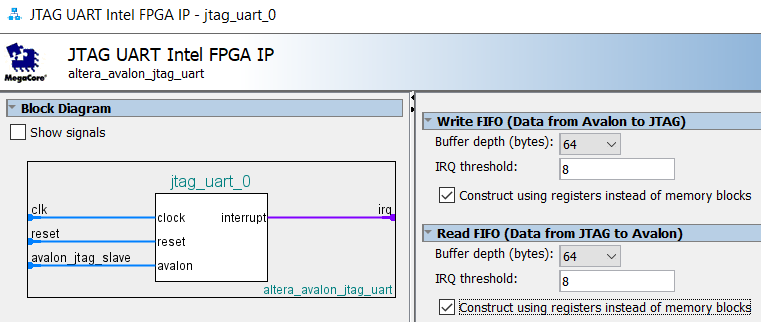


Рис. 2.7. Модуль JTAG UART.

Также необходимо добавить сам процессор с обычными настройками и следующими настройками вектора ошибок и сброса:

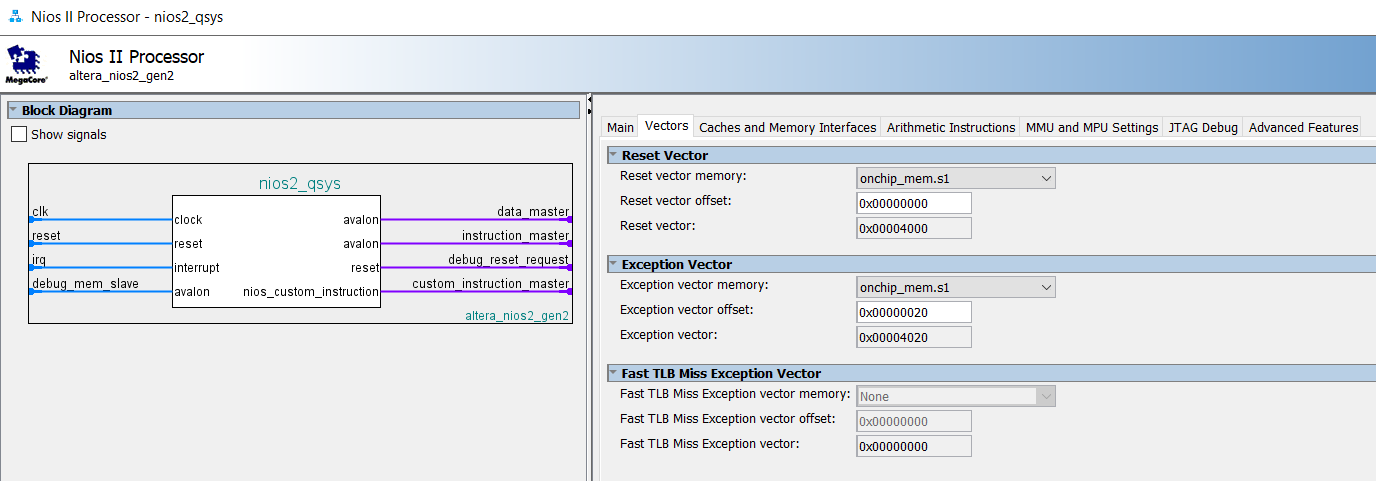


Рис. 2.8. Настройки вектора для Nios II.

Выполним соединения в Platform Designer созданных модулей следующим образом:

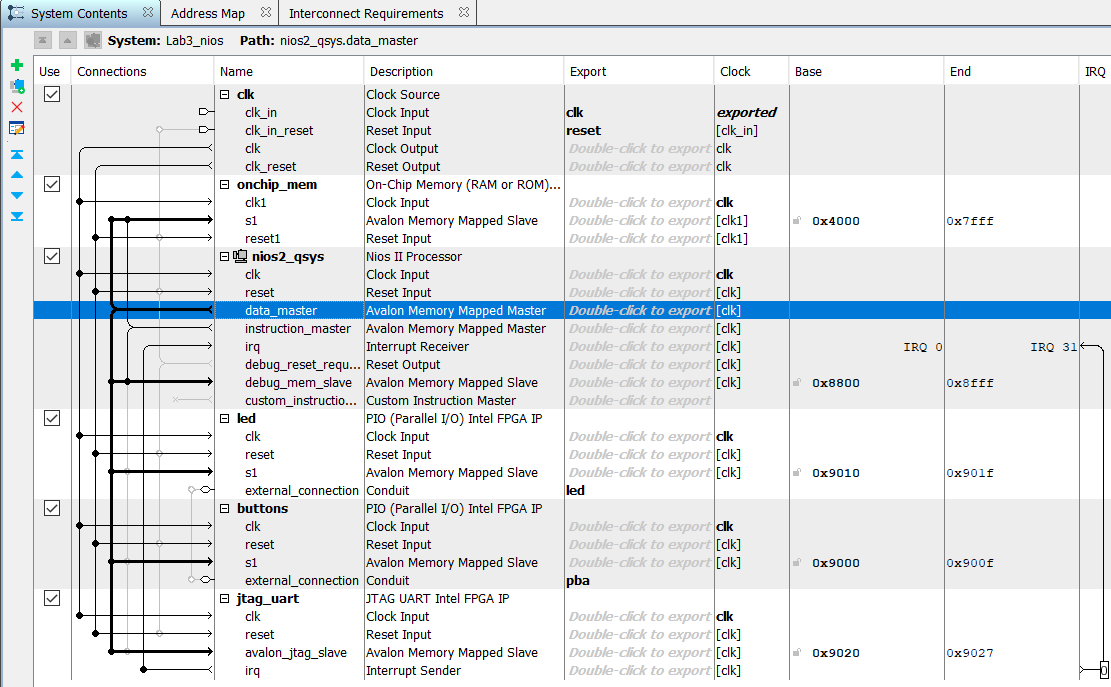


Рис. 2.9. Platform Designer.

Посмотрим, как выглядит Schematic:

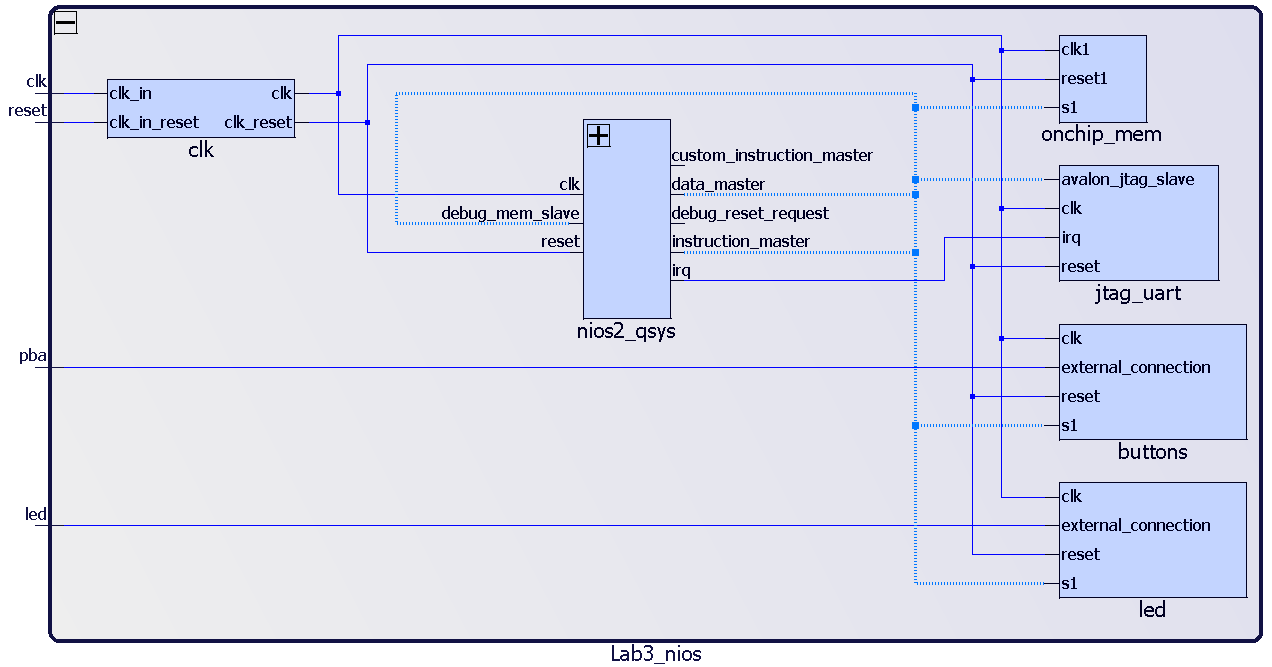


Рис. 2.10. Schematic в Platform Designer.

Как мы видим разработанная схема похожа на Рис. 2.1., что свидетельствует о корректности разработанного устройства.

Посмотрим на Interconnections, которые были добавлены в разработанной схеме:

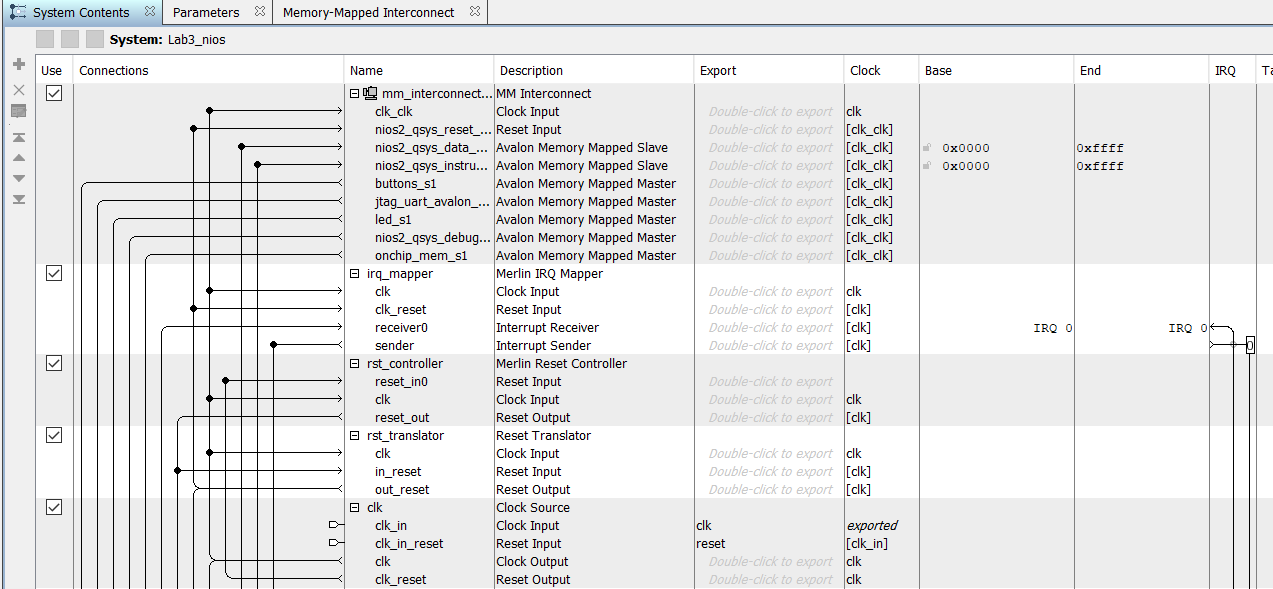


Рис. 2.11. System Interconnections.

Как мы видим, появилось 4 новых модуля.

rst\_controller служит для сброса процессора и памяти, а также передает сигнал сброса в rst\_translator, а он в свою очередь занимается сбросом периферии.

irq\_mapper занимается, как понятно из названия, прерываниями.

mm\_interconnect\_0 отвечает за передачу данных между периферией и процессором.

Выполним генерацию разработанного устройства и создадим файл верхнего уровня:



Выполним компиляцию и посмотрим на результат в RTL Viewer:

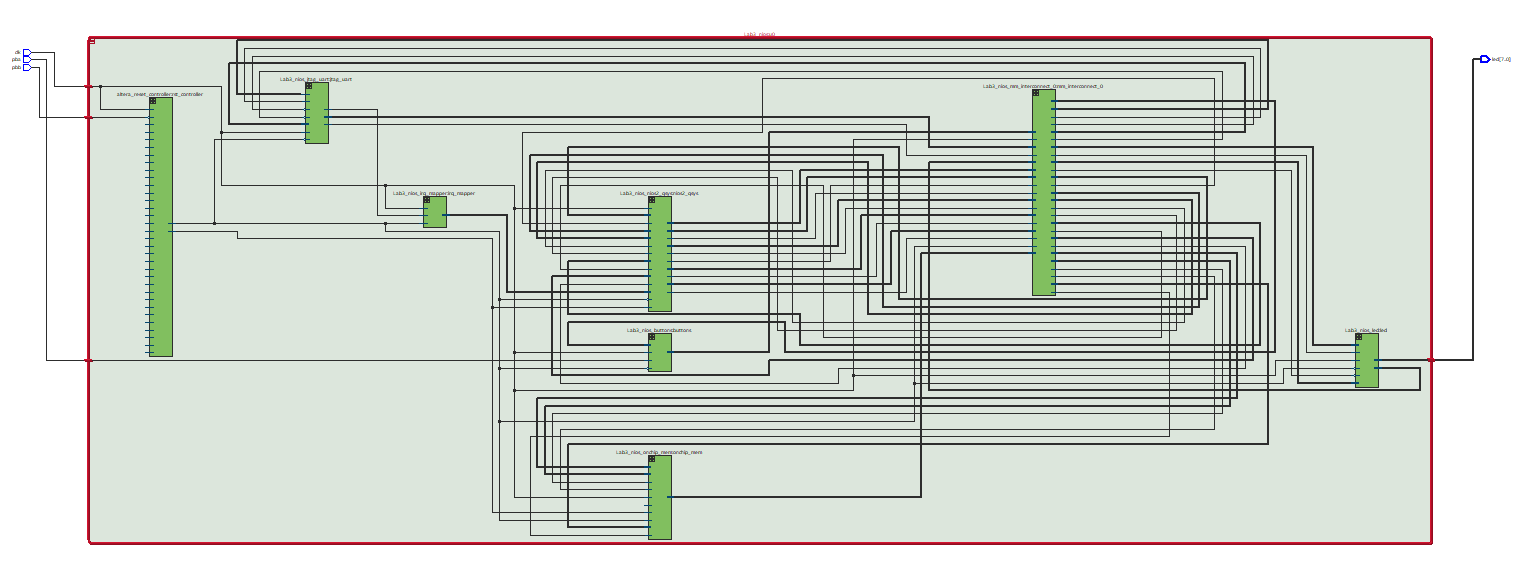


Рис. 2.12. RTL Viewer.

Данная схема аналогична приведенной в методических материалах, что свидетельствует о корректности разработанного устройства.

Далее выполним назначения входов и выходов в Pin Planner:

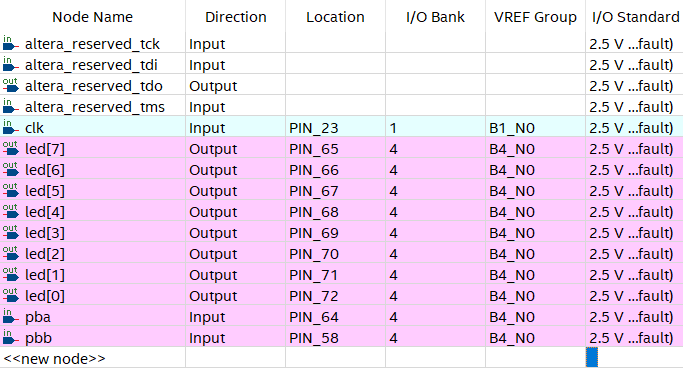


Рис. 2.13. Pin Planner.

Также добавим .sdc файл с временными требованиями:



Проведем полную компиляцию и проверим, что все они выполняются:

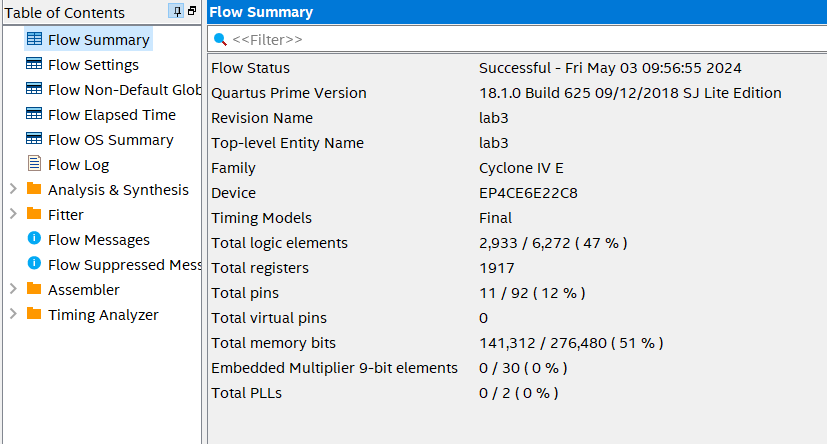


Рис. 2.14. Результат компиляции.

Как можно видеть, компиляция прошла успешно и временные требования выполняются.

Перейдем к разработке программа для Nios II:

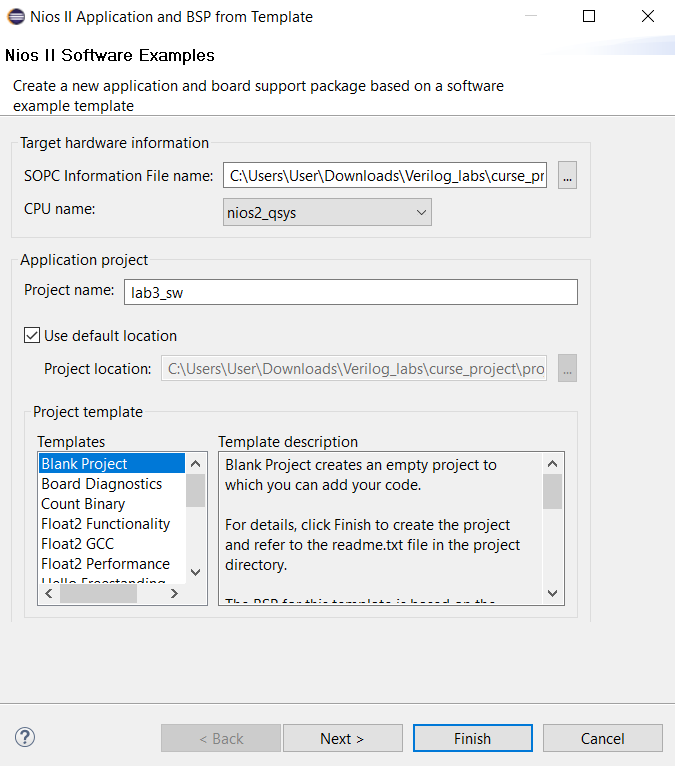


Рис. 2.15. Создание проекта для Nios II.

Создадим файл с исходным кодом:

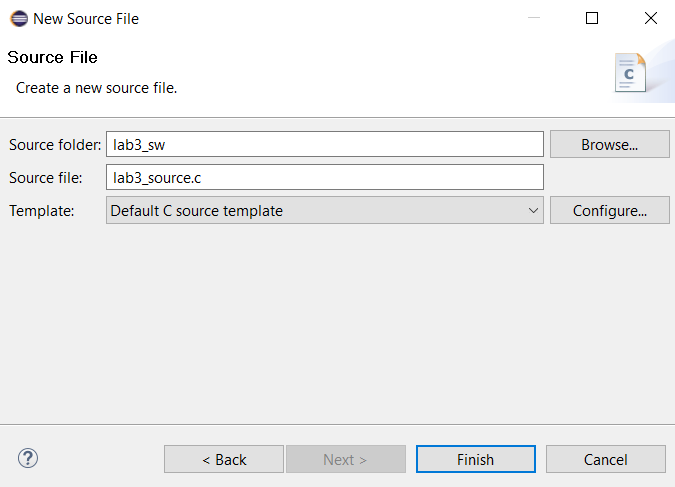


Рис. 2.16. Создание файла с исходным кодом.

Код приведен ниже:



При старте в консоль выводятся сообщения, а все светодиоды зажигаются (0 – активный).

Далее в цикле считывается значение с кнопки до тех пор, пока она не будет нажата. В случае нажатия происходит циклический сдвиг 1 в переменной, и она выводится на светодиоды в инвертированном виде, таким образом будет гореть только тот светодиод, которому соответствовала единица в переменной led.

После этого включается задержка, чтоб избежать дребезга.

Выполним следующие настройки для BSP, чтоб уменьшить объем проекта:

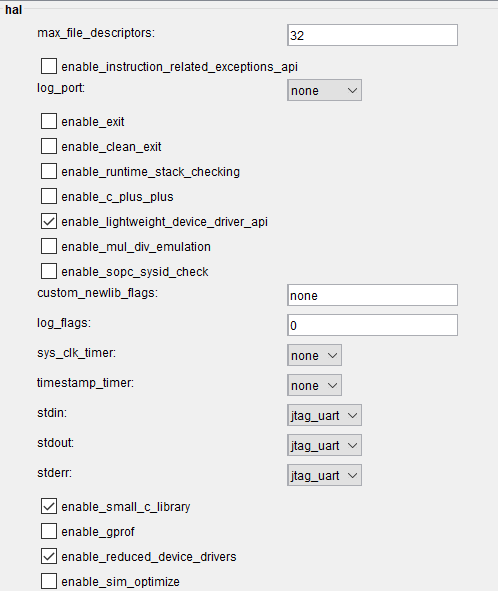


Рис. 2.17. Настройки BSP.

И выполним компиляцию:

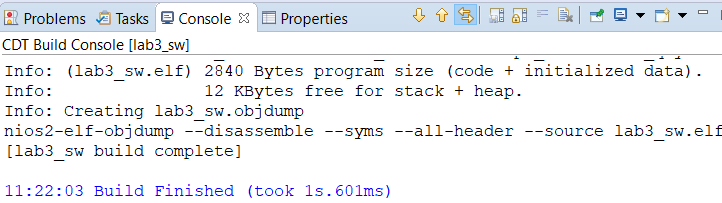


Рис. 2.18. Компиляция программы.

Компиляция прошла успешно.

Теперь запишем проект на плату и посмотрим на результат в консоли:

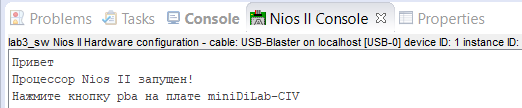


Рис. 2.19. Приветственные сообщения после запуска.

Как мы видим, после запуска действительно появились приветственные сообщения. Выполним несколько нажатий на pba:

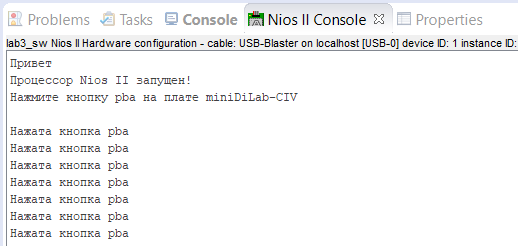


Рис. 2.20. Консоль после нескольких нажатий на pba.

После каждого нажатия появляется надпись, а также светодиод сдвигается левее. По достижению led8 он переходит к led1, как и было задумано.

Теперь выполним анализ размера файла после компиляции при различных настройках.

Поставим следующие настройки для BSP:

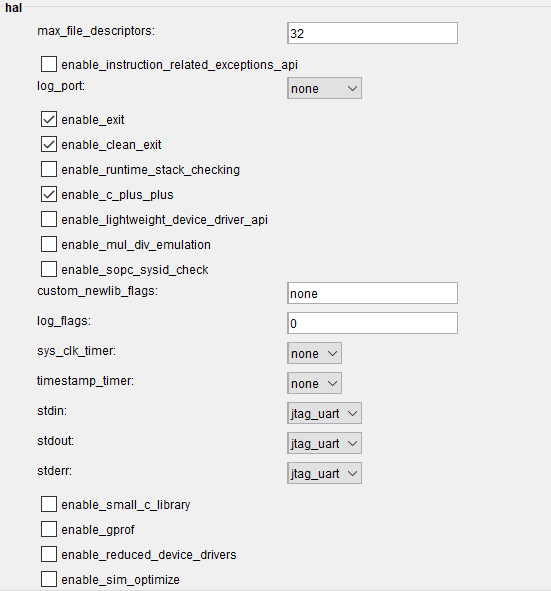
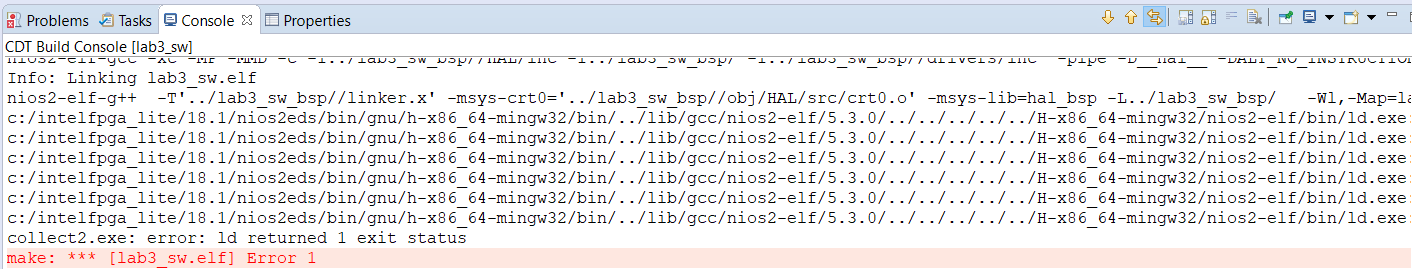


Рис. 2.21. Новые настройки для BSP.

Выполним генерацию этого BSP и компиляцию проекта:



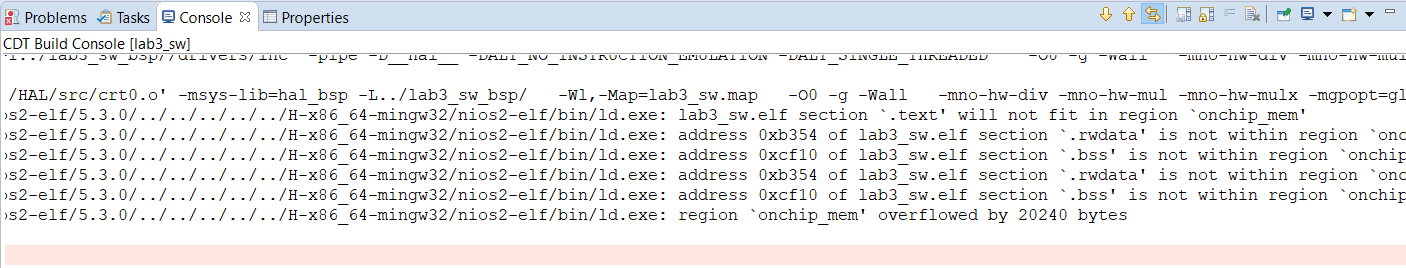
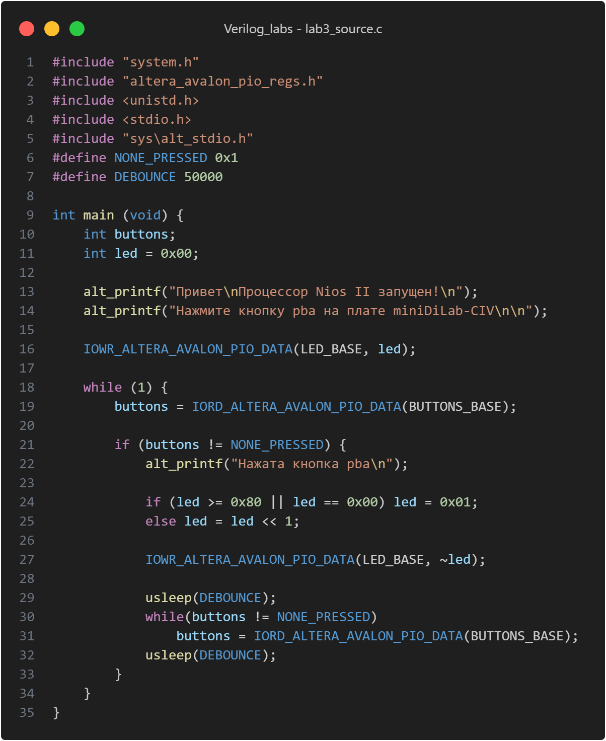


Рис. 2.22. Компиляция проекта с новыми настройками BSP.

Видим, что для компиляции не хватило памяти устройства, а именно 20240 байт.

Вернем настройки в прошлую форму, чтоб файл мог поместиться в выделенной памяти.

Проексперементируем теперь с кодом, заменим printf на более легкие alt\_printf. Они более ограничены в функционали, однако для задачи вывода текста на экран вполне подходят:



Выполним компиляцию и получим следующий результат:

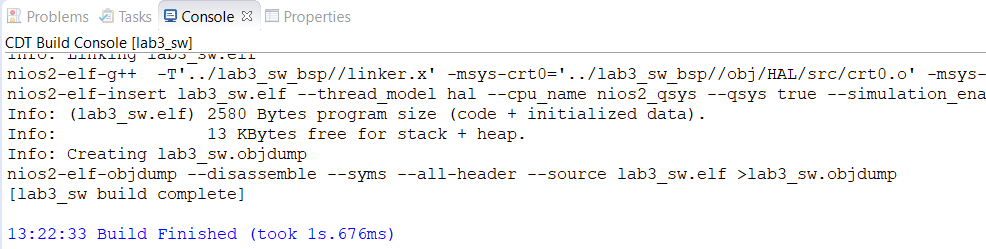


Рис. 2.23. Результат компиляции.

Как мы видим, размер файла стал меньше, как и ожидалось.

Попробуем еще уменьшить размер файла, выставим настройки оптимизации по размеру файла:

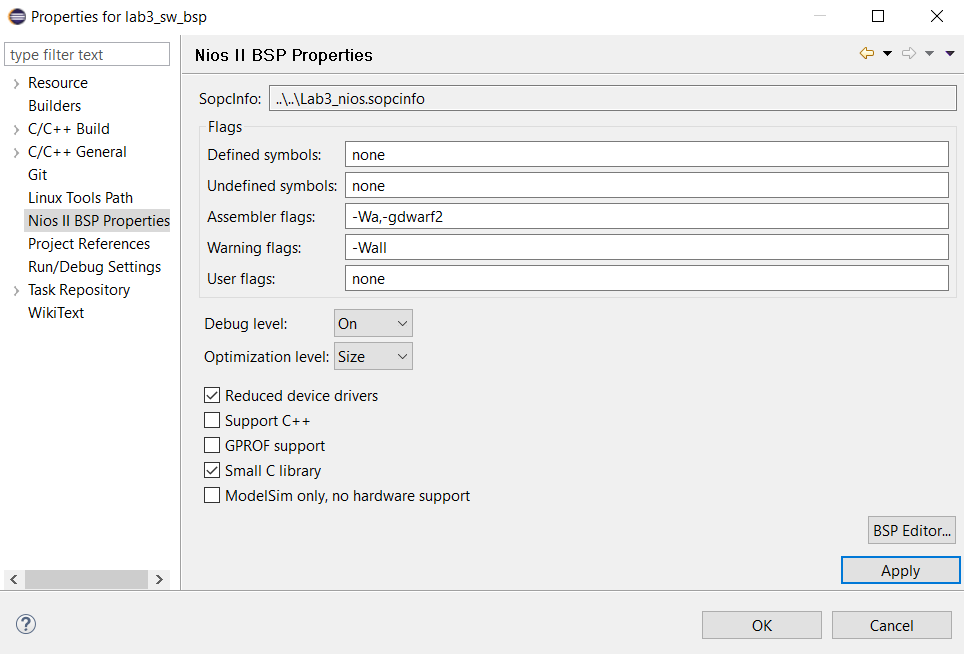


Рис. 2.24. Optimization level: size. У lab3\_sw\_bsp.

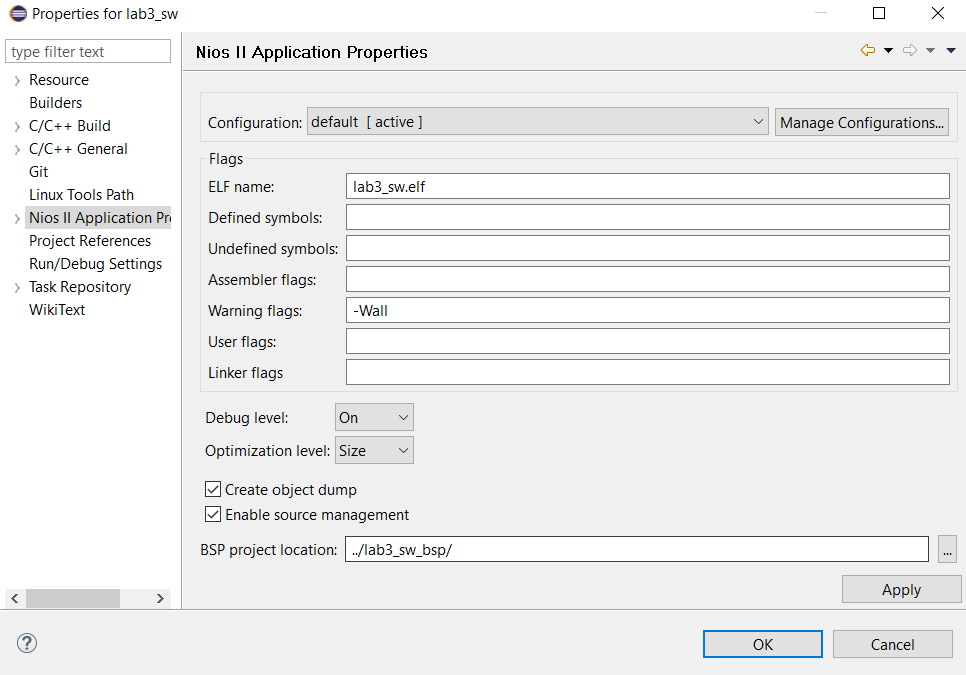


Рис. 2.25. Optimization level: size. У lab3\_sw.

Повторим компиляцию:

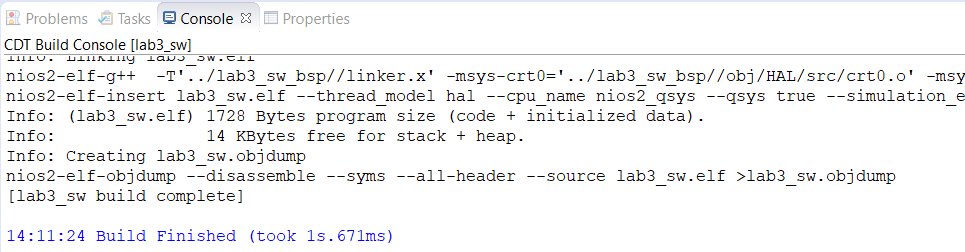


Рис. 2.26. Компиляция после выставления настроек оптимизации.

Видим, что размер файла уменьшился еще сильнее, до 1728 байт.

Повторно запрограммируем плату и убедимся, что результат выполнения не изменился:

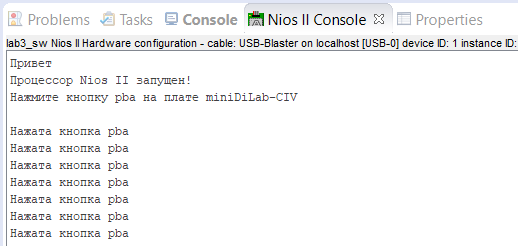


Рис. 2.27. Результат запуска программы и нескольких переключений pba.

Как мы видим, программа работает так же, как и до оптимизаций.

# Проект 2:

## Структура проекта:

Структура разрабатываемого проекта приведена ниже:

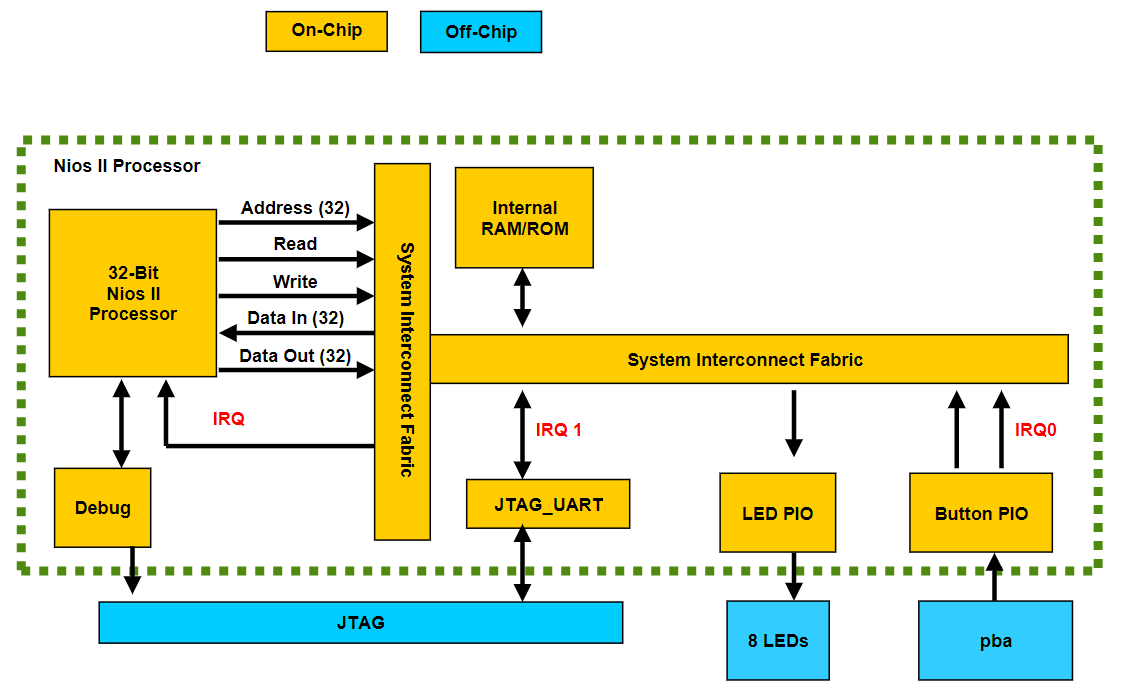


Рис. 3.1. Структура разрабатываемого проекта.

Под управлением процессора NIOSII обеспечивается:

* Работа по прерываниям от нажатия кнопки pba.
* Формирование на консоли сообщений о нажатой кнопке.
* При каждом нажатии кнопки pba происходит изменение номера включенного светодиода от led1 к led8 на одну позицию (с циклическим переходом от led8 к led1).

## Решение:

Выполним создание проекта в QP со следующими настройками:

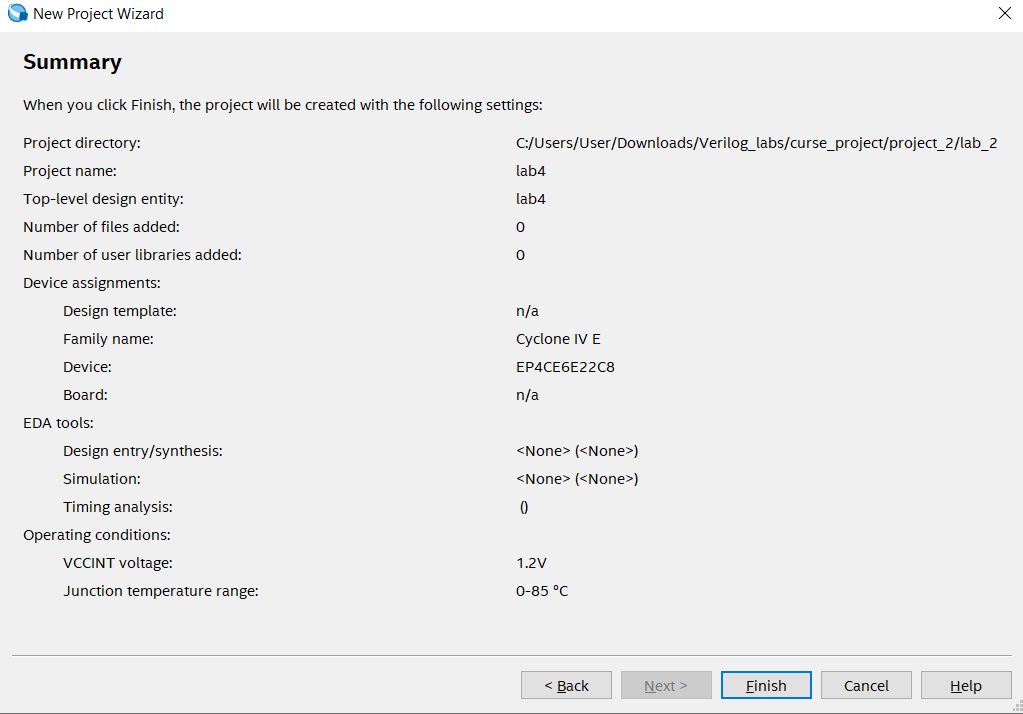


Рис. 3.2. Настройки для QP.

Далее переходим в Platform Designer. Скопируем PD из Проект 1.

Для того, чтоб мы могли работать с кнопками по прерываниям необходимо выполнить следующие настройки c модулем PIO:

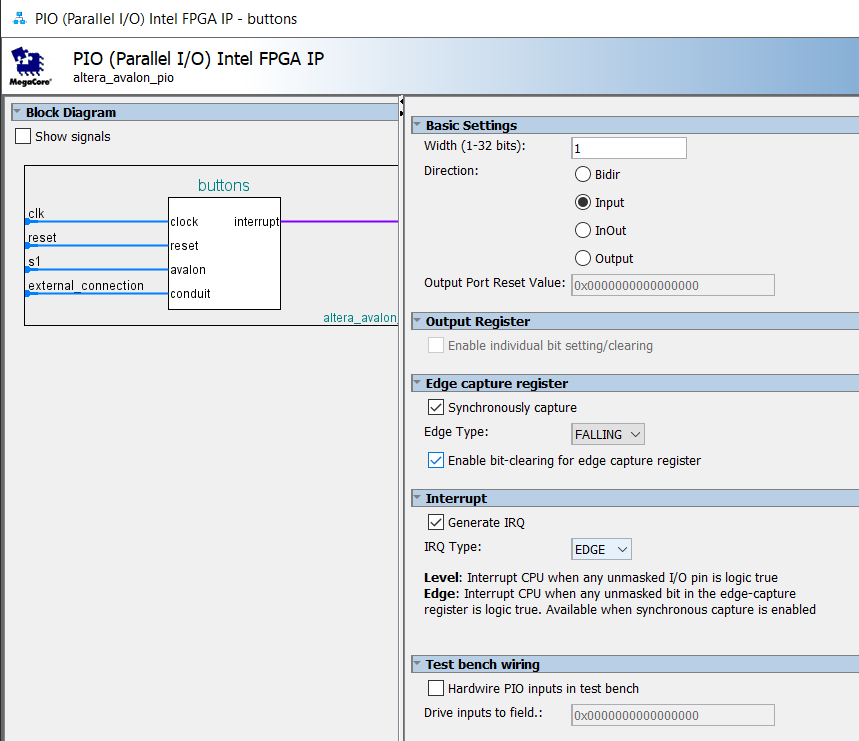


Рис. 3.3. Новые настройки модуля PIO.

Также необходимо подключить прерывания к процессору:

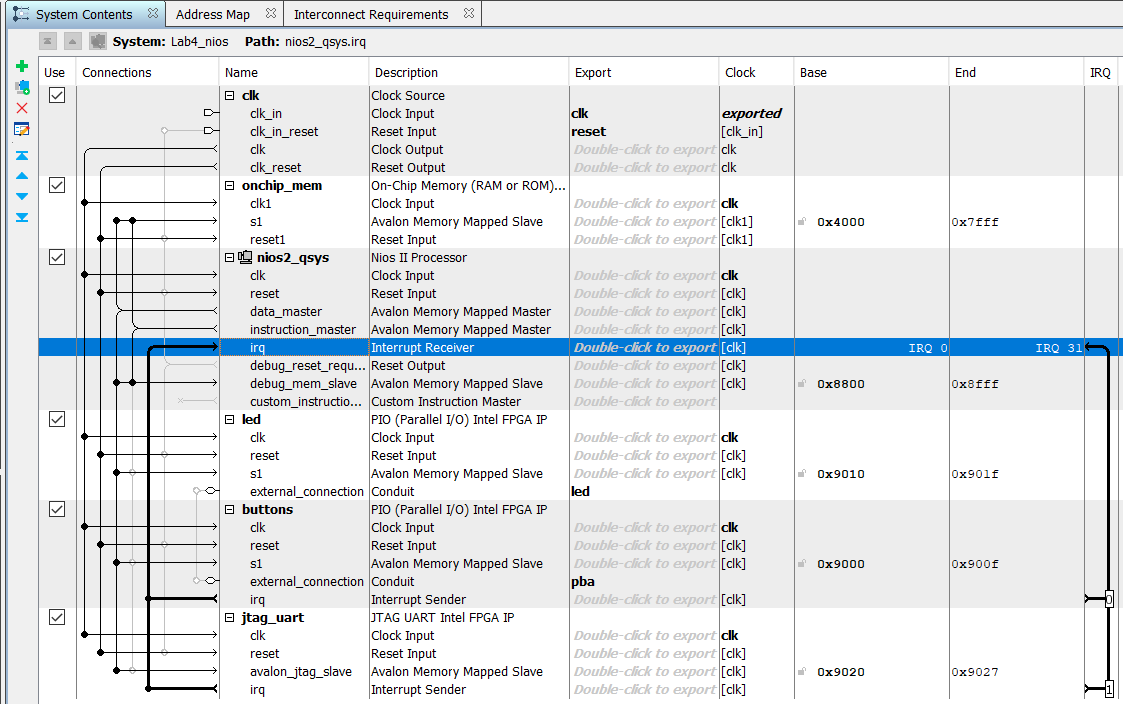


Рис. 3.4. Подключение прерываний к процессору.

Выполним генерацию модуля и создадим файл верхнего уровня:



Выполним компиляцию и выполним назначения для входов и выходов:

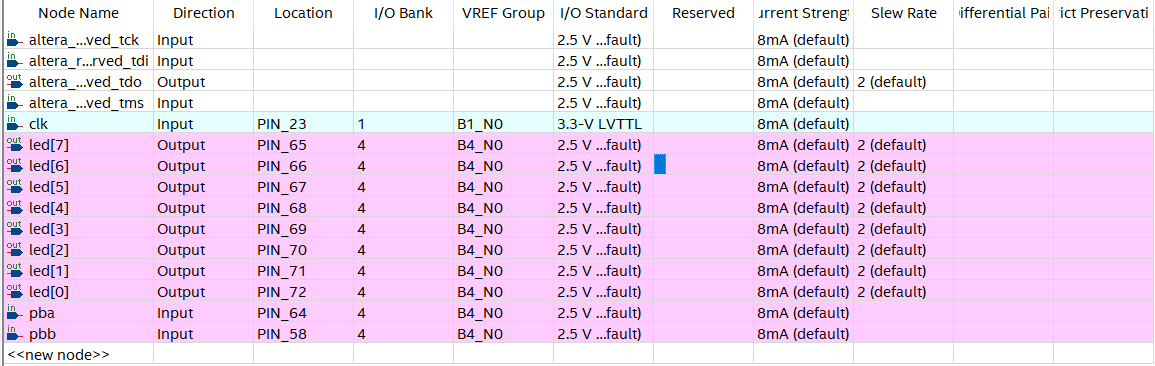


Рис. 3.5. Назначения в Pin Planner.

Также добавим файл для временных характеристик:



Выполним полную компиляцию и проверим, выполняются ли временные характеристики:

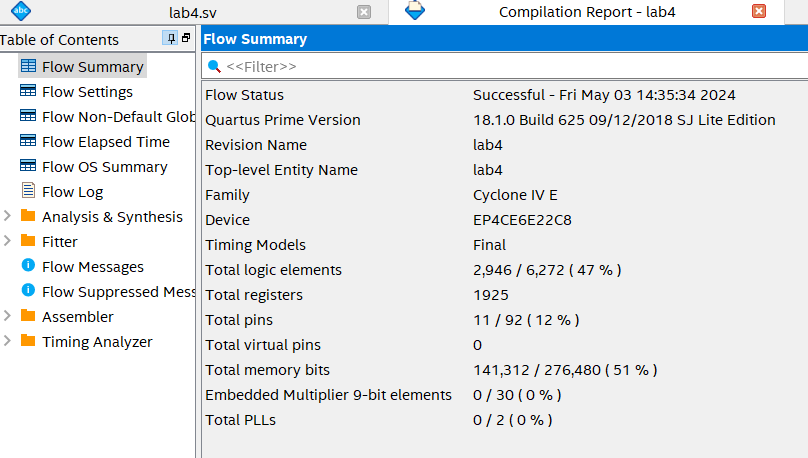
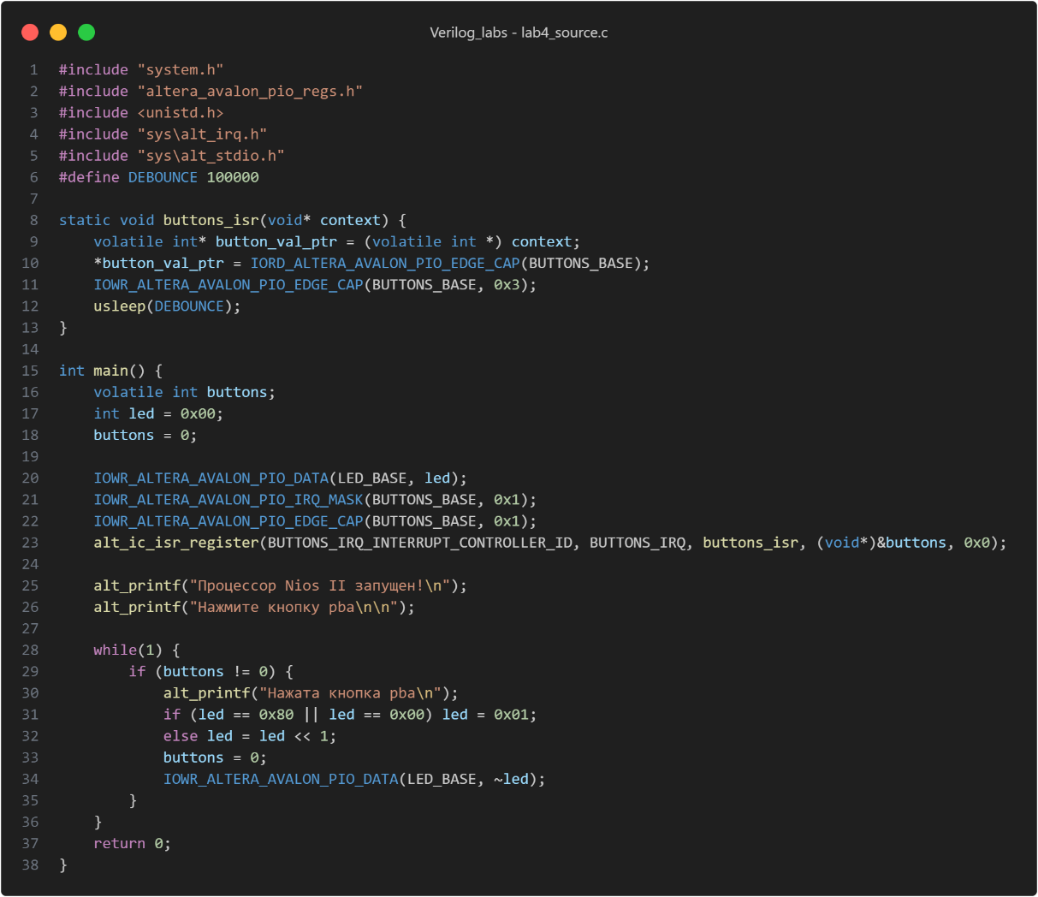


Рис. 3.6. Результат компиляции.

Как видим все временные характеристики выполняются.

Теперь перейдем к созданию проекта Nios II. Сам проект создается с настройками, рассмотренными в Проект 1. Текст программы приведен ниже:



Функция buttons\_isr должна вызываться при нажатии кнопки. Она записывает 1 при нажатии в переданную переменную, после этого она очищает все позиции захвата и запускает паузу в этой функции.

Основная функция зажигает све светодиоды, включает прерывания для кнопки, сбрасывает позицию захвата и регестрирует функцию, которую вызывать при прерывании.

Дадее уходим в бесконечный цикл, когда срабатывает прерывание значение переменной buttons устанавливается в 1 и мы реагируем на это, сдвигая заженный светодиод.

Выполним компиляцию:

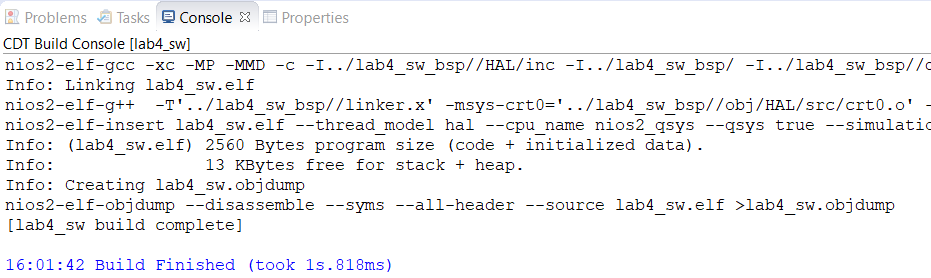


Рис. 3.7. Результат компиляции.

Как видим размер проекта равен 2560 байт. Запишем его на плату, тогда в консоли увидим:

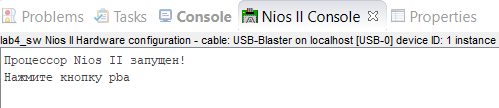


Рис. 3.8. Консоль проекта после записи программы.

Нажмем кнопку несколько раз, будет следующий результат:

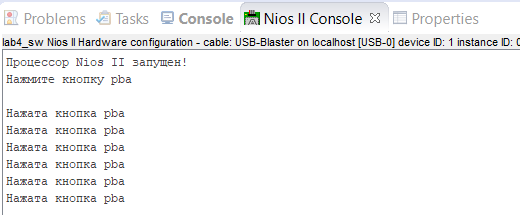


Рис. 3.9. Консоль после нескольких запусков программы.

Как мы видим программа работает корректно в соответствии с ожиданиями.

# Вывод:

В ходе курсовой работы были изучены различные возможности процессора Nios II.

В первой программе были получены навыки по оптимизации программного кода и уменьшению его объема, занимаемого на кристалле, что позволит делать комплексные проекты тратя минимальные ресурсы на память.

Во втором проекте мы поработали с прерываниями, они позволяют эффективно обрабатывать запросы пользователей к программе.