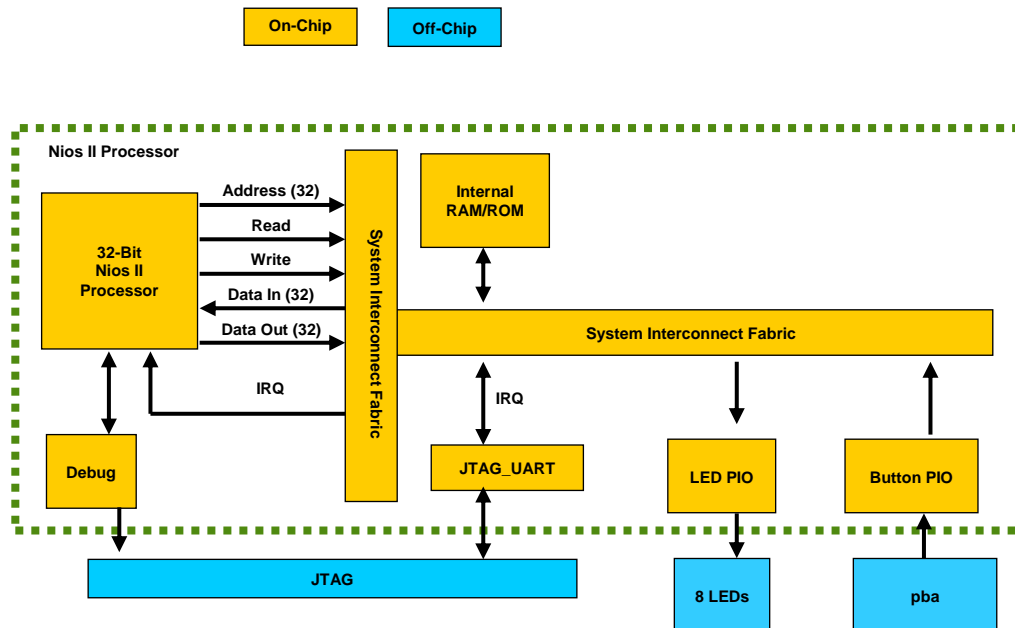


Задание labn_3

Введение:

Цель упражнения – расширить знакомство с возможностями по реализации и отладки проектов на базе процессора NIOSII

Структура проекта



Алгоритм работы проекта:

Под управлением процессора NIOSII обеспечивается:

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

Часть 1 – Создание проекта

1. Запустите пакет QuartusII
2. В меню **File** менеджера пакета, укажите **New Project Wizard...**
3. На экране появится окно введения - **Introduction** (если оно небыло отключено). Нажмите кнопку **next**.
4. В появившемся окне введите следующие данные:

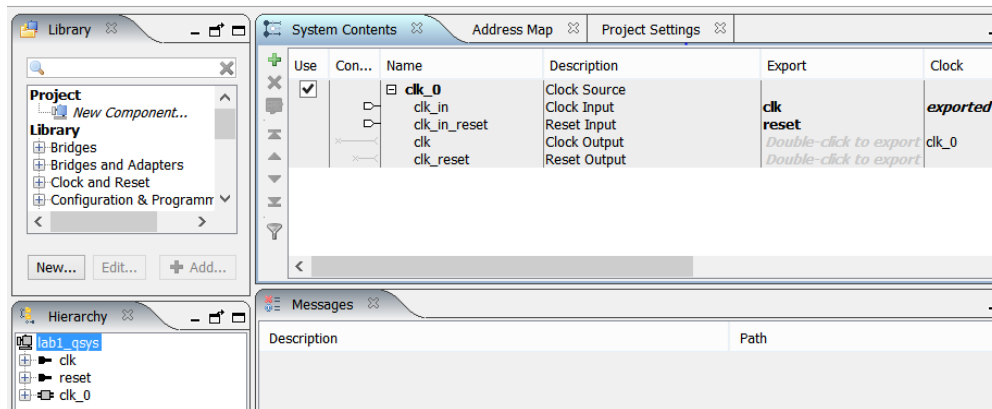
What is the working directory for this project? Рабочая папка (с помощью браузера найдите рабочую папку проекта)	Папка инсталляции\labs\lab3\
What is the name of this project? Имя проекта	lab3
What is the name of the top-level design entity for this project? Имя модуля верхнего уровня в иерархии проекта.	lab3

5. Нажмите кнопку **Next**.
6. В окне **Add Files [page 2 of 5]** нажмите кнопку **Next**.
7. В окне **Family & Device Setting [page 3 of 5]**:
 - в разделе **Family** укажите **Cyclone IV E**.
 - в разделе **Available devices** укажите СБИС **EP4CE6E22C8**.
 Нажмите кнопку **Next**.
8. В окне **EDA Tool Setting [page 4 of 5]** оставьте все без изменения и нажмите кнопку **Next**.
9. Появится окно **Summary [page 5 of 5]**, в котором указаны установки, заданные Вами для создаваемого проекта. Проверьте их. Если все правильно, то нажмите кнопку **Finish**. В противном случае, вернитесь назад, нажав (возможно несколько раз) кнопку **Back**.

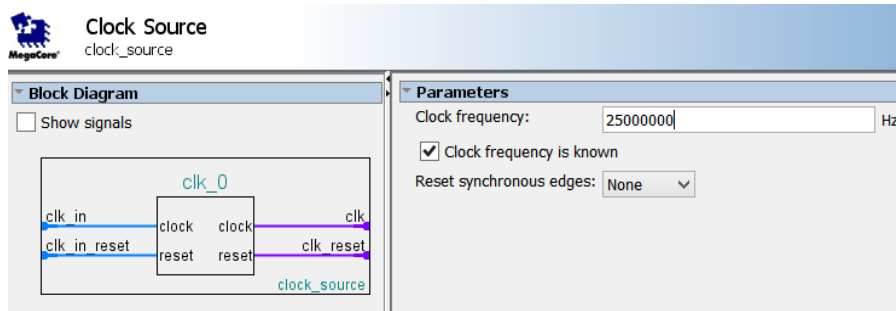
Проект создан.

Часть 2 – Создание аппаратной части проекта

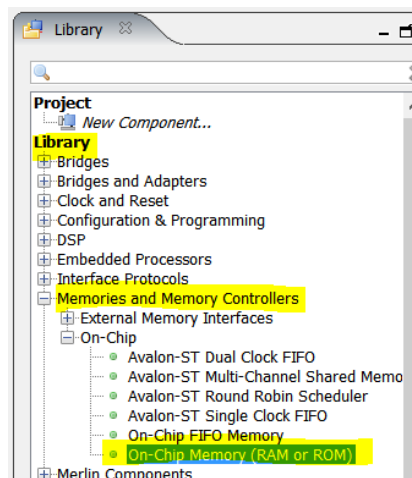
1. Выполните команду **Tools => Qsys**. Будет запущен Qsys и откроется закладка System Contents, в которую по умолчанию будет добавлен компонент source clock



2. Выполните команду **File=>Save as** и сохраните систему под именем **lab2_qsys**
3. Дважды щелкните по модулю clock source => откроется окно задания параметров модуля. Задайте частоту тактового сигнала = 25 МГц, что соответствует частоте кварцевого генератора на плате miniDiLaB-CIV. Нажмите кнопку Finish.



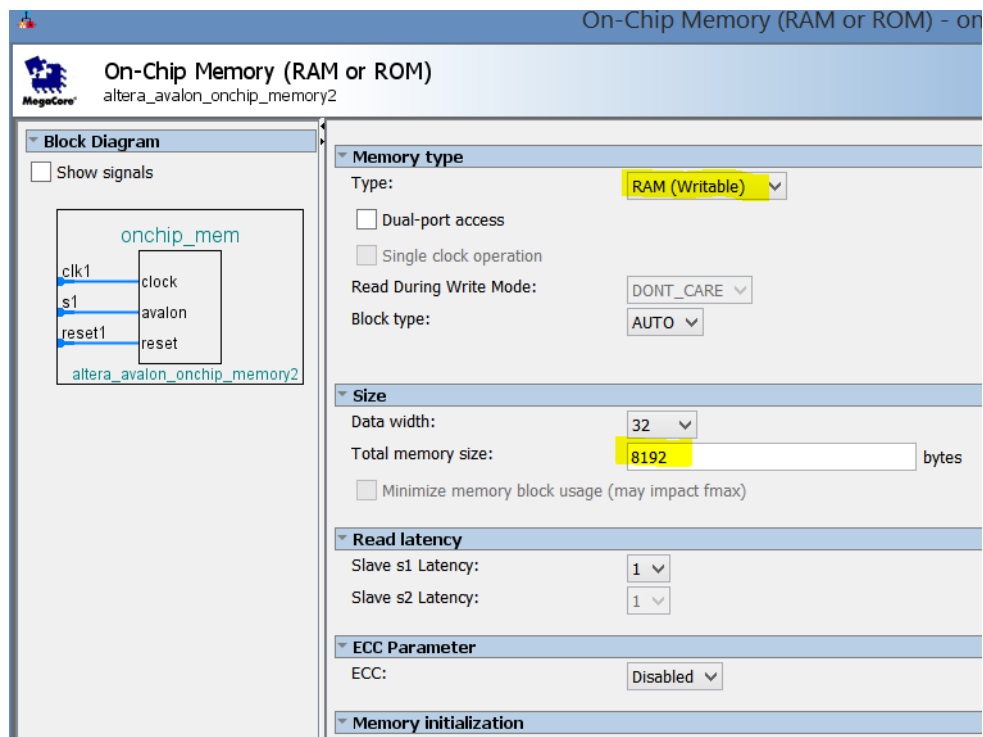
- ✓ Переименуйте компонент: в закладке System Contents выберите имя компонента, нажмите правую клавишу мыши и выберите команду Rename. Новое имя clk.
4. Создание, на основе встроенных модулей M9K, памяти для команд и данных процессора
- ✓ В списке доступных компонентов



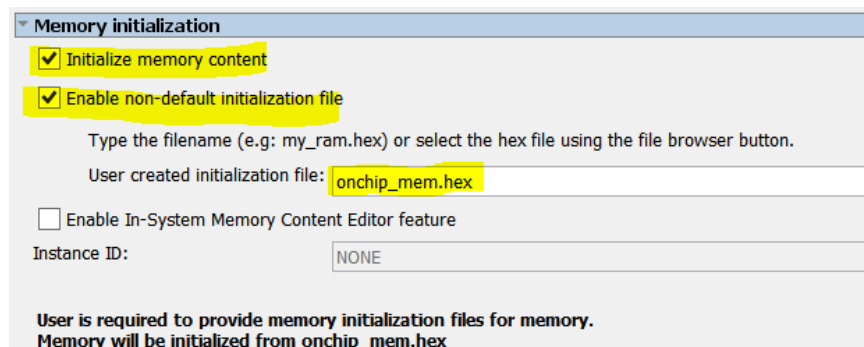
найдите компонент **On-Chip Memory (RAM and ROM)** и дважды щелкните левой клавишей мыши.

- ✓ Появится окно On-Chip Memory (RAM and ROM)

- ✓ В разделе memory type задайте тип памяти – RAM
- ✓ В разделе Size задайте размер памяти 8192 байт.
- ✓ В разделе memory Initialization оставьте все значения по умолчанию – автоматически будет создан файл инициализации памяти, имя которого указано внизу страницы с настройками



- ✓ Нажмите кнопку Finish. Память для команд и данных процессора создана.
- ✓ Так как, для нормальной работы компонент должен быть подключен к тактовому сигналу, сигналу сброса внутренних регистров и Мастеру на шине Avalon-MM, то появятся сообщения об ошибках и предупреждение. Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Переименуйте созданный модуль памяти: в закладке System Contents выберите имя созданного модуля памяти, нажмите правую клавишу мыши и выберите команду Rename. Новое имя onchip_mem.
- ✓ Двойным щелчком в поле компонента откройте окно настройки и установите опцию Enable non-default initialization file (имя файла д.б. onchip_mem.hex)

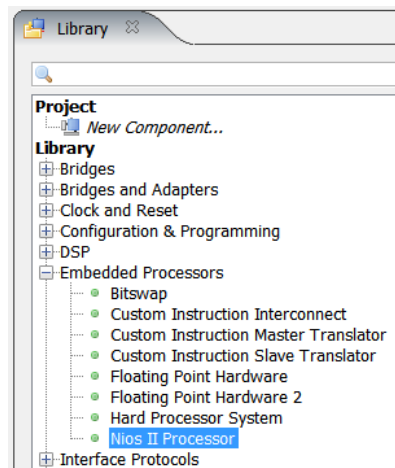


- ✓ Соедините выход clk компонента clk с входом clk1 компонента onchip_mem, а выход clk_reset компонента clk source с входом reset1 компонента onchip_mem.

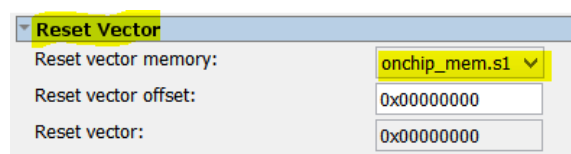
System Contents						
Use	Con...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		clk	Clock Source			
		clk_in	Clock Input			
		clk_in_reset	Reset Input			
		clk	Clock Output			
		clk_reset	Reset Output			
<input checked="" type="checkbox"/>		onchip_mem	On-Chip Memory (RAM or ROM)			
		clk1	Clock Input			
		s1	Avalon Memory Mapped Slave			
		reset1	Reset Input			

5. Конфигурация и подключение к системе ядра процессора NIOSII

- ✓ В списке доступных компонентов выберите раздел Processors => NIOSII Processor и дважды щелкните левой клавишей мыши.



- ✓ Появится окно конфигурации процессора.
- ✓ На закладке Core NIOSII установите:
 - тип процессора - NIOSII/e –(простейший вариант процессорного ядра)
- ✓ Переключитесь на закладку JTAG Debug Module и установите параметр Select a debugging level соответствующим **Level 1**.
- ✓ Нажмите кнопку Finish. Ядро процессорного модуля создано и включено в систему.
- ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Переименуйте созданный процессорный модуль: в закладке System Contents выберите имя созданного модуля, нажмите правую клавишу мыши и выберите команду Rename. Новое имя nios2_qsys.
- ✓ Соедините вход clk компонента nios2_qsys с выходом clk1 компонента clk, а выход clk_reset компонента clk с входом reset_n компонента nios2_qsys.
- ✓ Соедините вход s1 компонента onchip_mem с выходами data_master и instruction_master компонента nios2_qsys.
- ✓ Откройте окно настройки процессора Nios: дважды щелкните левой клавишей мыши по компоненту nios2_qsys.
- ✓ На закладке Core NiosII укажите память для вектора сброса:



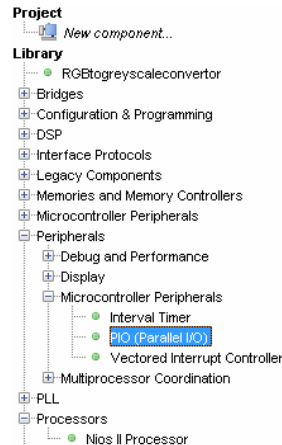
- ✓ На закладке Core NiosII укажите память для вектора exception:

Exception Vector	
Exception vector memory:	onchip_mem.s1
Exception vector offset:	0x00000020
Exception vector:	0x00000020

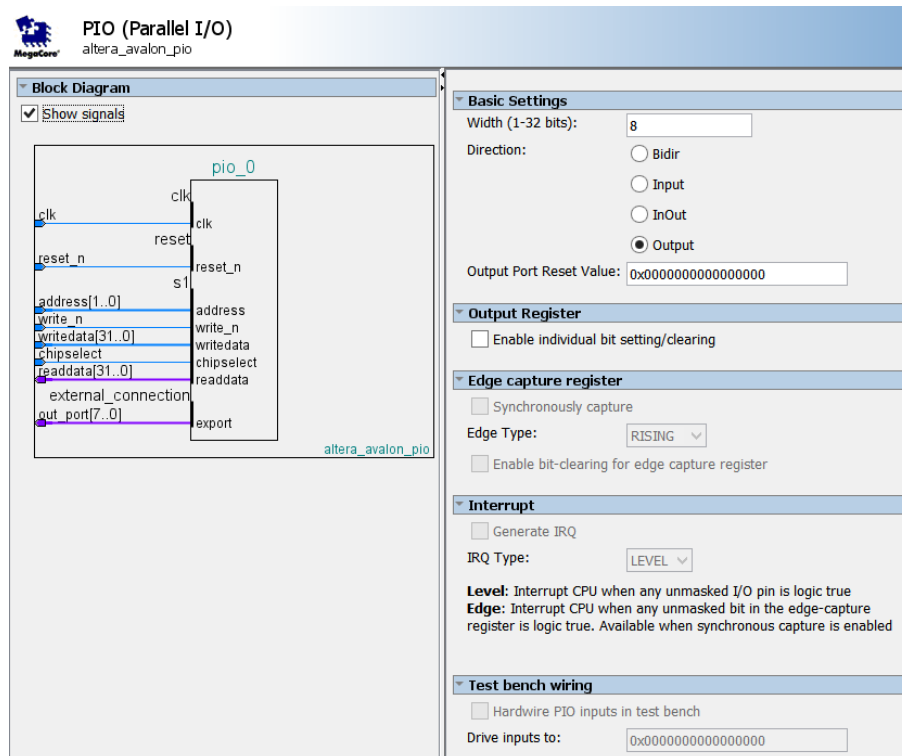
- ✓ Нажмите кнопку Finish.

6. Конфигурация и подключение к системе модуля PIO (параллельного ввода вывода).

- ✓ В списке доступных компонентов выберите раздел PIO (Parallel I/O) и дважды щелкните левой клавишей мыши.



- ✓ Откроется окно настройки.
- ✓ Откройте закладку Basic Settings и установите следующие параметры:
 - Разрядность width = 8
 - Направление передачи – Direction = Output.
 - Reset value=0;

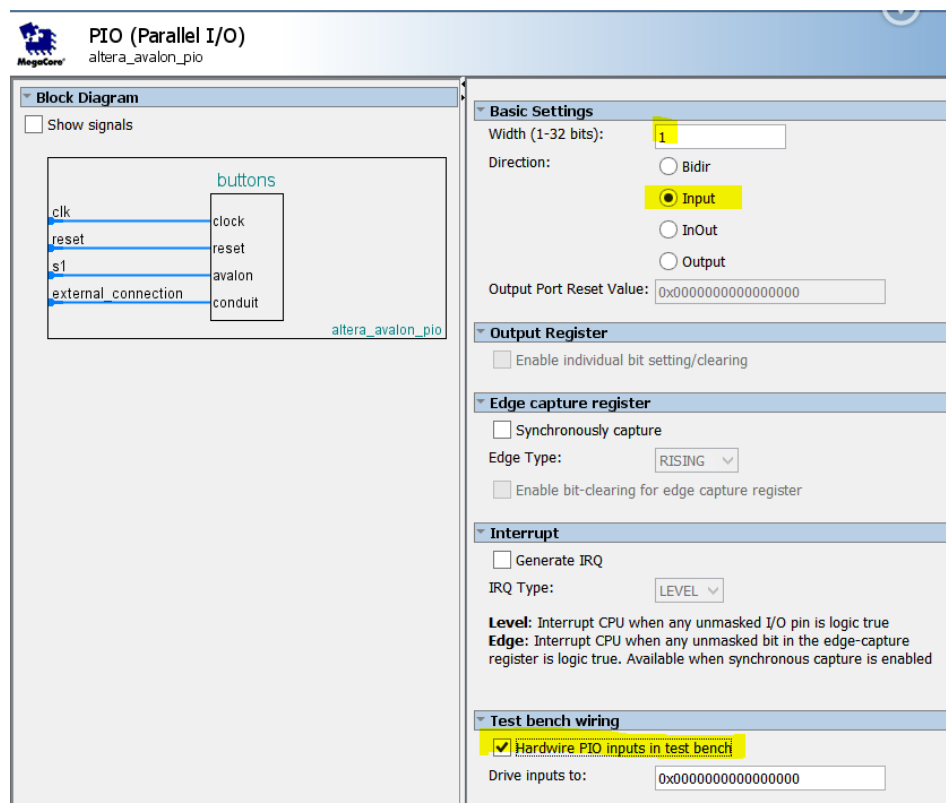


- ✓ Нажмите кнопку Finish.

- ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Переименуйте созданный компонент: в закладке System Contents выберите имя созданного компонента, нажмите правую клавишу мыши и выберите команду Rename. Новое имя led.
- ✓ Соедините вход clk компонента led с выходом clk1 компонента clk, а выход clk_reset компонента clk с входом reset компонента pio.
- ✓ Соедините вход s1 компонента led с выходами data_master компонента nios2_qsys.
- ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Дважды щелкните в строке external_connection столбца Export, введите имя внешнего вывода создаваемой системы - led
- ✓ Модуль PIO настроен и подсоединен к системе.

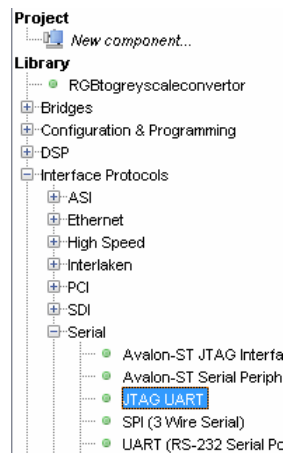
7. Конфигурация и подключение к системе еще одного модуля PIO (параллельного ввода вывода).

- ✓ В списке доступных компонентов выберите раздел PIO (Parallel I/O) и дважды щелкните левой клавишей мыши.
- ✓ Откроется окно настройки.
- ✓ Откройте закладку Basic Settings и установите следующие параметры:
 - Разрядность width = 1
 - Направление передачи – Direction = Input.

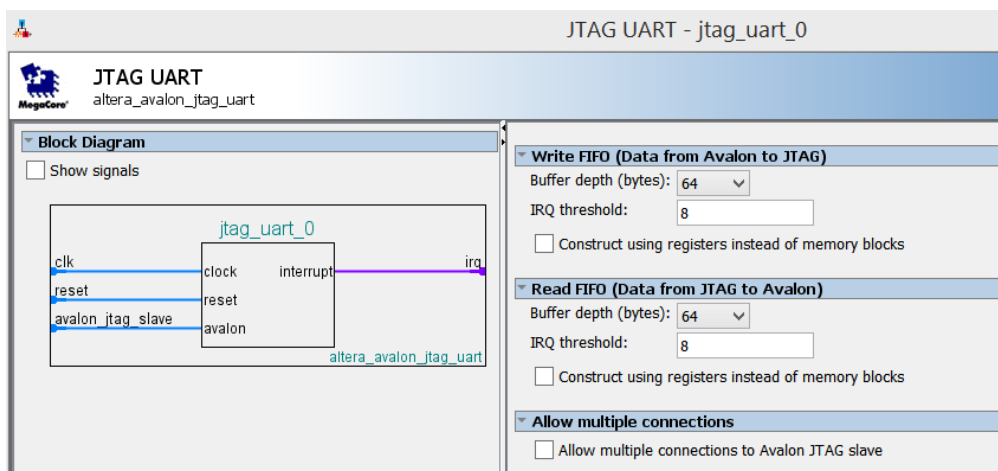


- ✓ Нажмите кнопку Finish.
- ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Переименуйте созданный компонент: в закладке System Contents выберите имя созданного компонента, нажмите правую клавишу мыши и выберите команду Rename. Новое имя buttons.
- ✓ Соедините вход clk компонента buttons с выходом clk1 компонента clk, а выход clk_reset компонента clk с входом reset компонента buttons.

- ✓ Соедините вход s1 компонента buttons с выходами data_master компонента nios2_qsys.
 - ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
 - ✓ Дважды щелкните в строке external_connection столбца Export, введите имя внешнего вывода создаваемой системы - pbb
 - ✓ Модуль PIO настроен и подсоединен к системе.
8. Конфигурация и подключение к системе модуля JTAG UART, используемого для передачи символьных данных между процессором и PC через JTAG (USB Blaster...). Будет использован как порт для стандартного ввода вывода процессора.
- ✓ В списке доступных компонентов выберите раздел JTAG UART и дважды щелкните левой клавишей мыши.

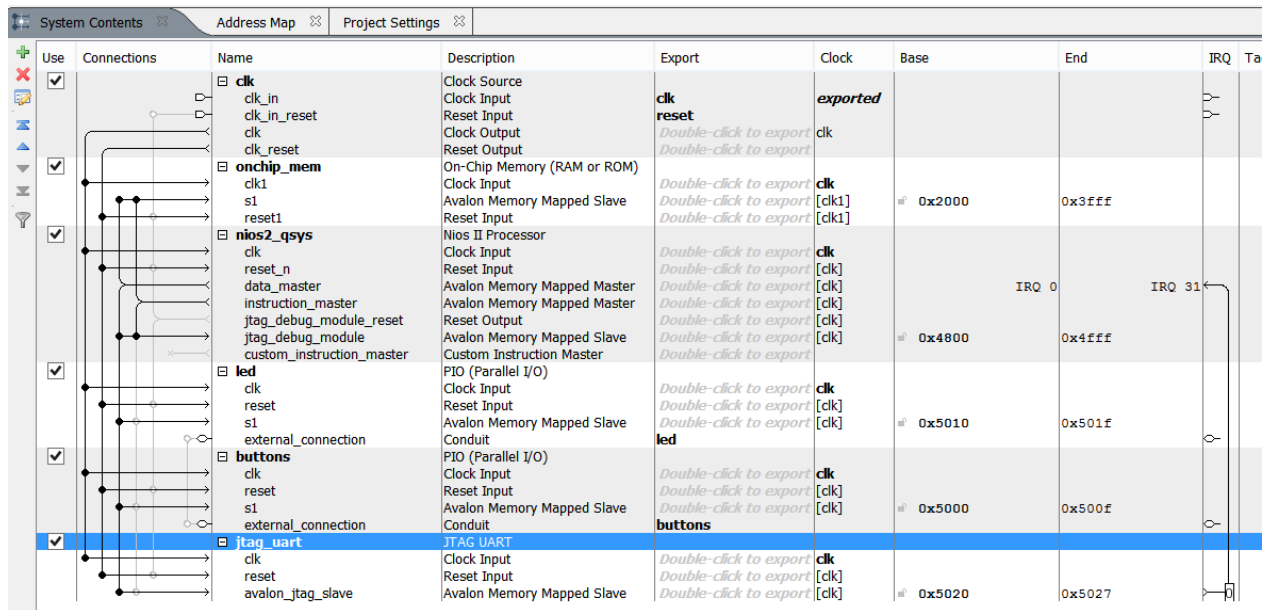


- ✓ Откроется окно настройки.
- ✓ Установите в нем значения приведенные ниже.






- ✓ Нажмите кнопку Finish.
- ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Переименуйте созданный компонент: в закладке System Contents выберите имя созданного компонента, нажмите правую клавишу мыши и выберите команду Rename. Новое имя jtag_uart.
- ✓ Соедините вход clk компонента jtag_uart с выходом clk1 компонента clk, а выход clk_reset компонента clk с входом reset компонента jtag_uart.
- ✓ Соедините вход avalon_jtag_slave компонента jtag_uart с выходами data_master компонента nios2_qsys.
- ✓ Соедините выход прерывания (колонок IRQ) компонента с входом прерывания компонента nios2_qsys

9. Выполните автоматическое распределение адресного пространства системы: System=>Assign base Addresses
10. Внешний вид созданной системы, закладка System Contents



- ## 11. Внешний вид созданной системы, закладка Address Map

System Contents  Address Map  Project Settings 

	nios2_qsys.data_master	nios2_qsys.instruction_master
onchip_mem.s1	0x2000 - 0x3fff	0x2000 - 0x3fff
nios2_qsys.jtag_debug_module	0x4800 - 0x4fff	0x4800 - 0x4fff
led.s1	0x5010 - 0x501f	
buttons.s1	0x5000 - 0x500f	
jtag_uart.avalon_jtag_slave	0x5020 - 0x5027	

- ## 12. Внешний вид созданной системы, закладка Project Settings

System Contents
Address Map
Project Settings

System

system

Device Settings

Device family: Cyclone IV E
Device: EP4CE6E22C8

Interconnect Settings

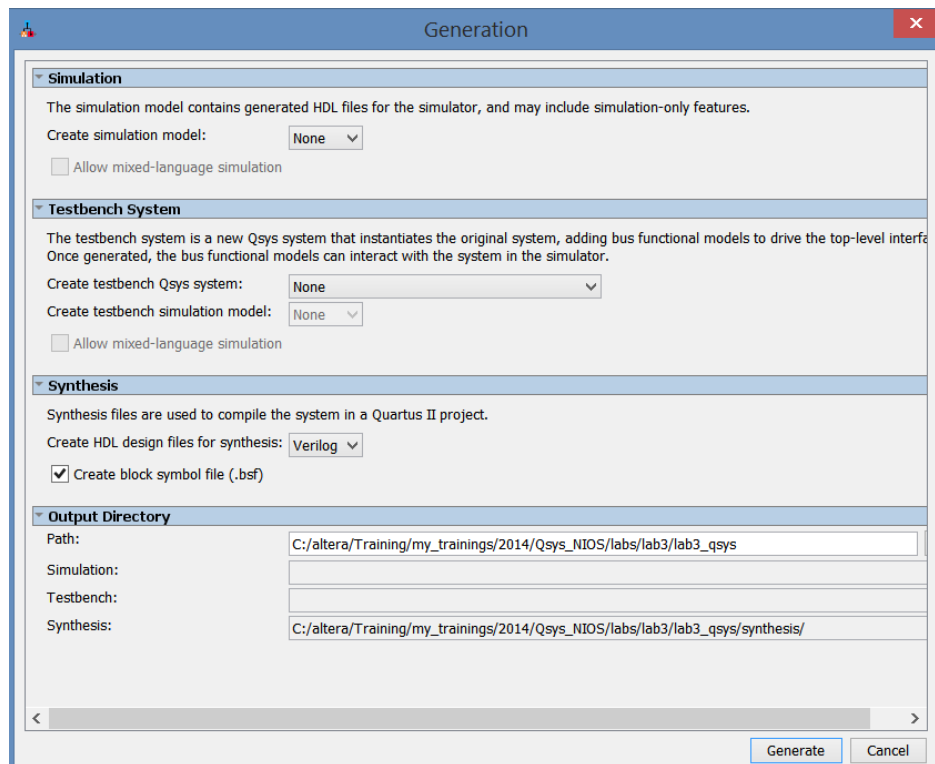
Clock crossing adapter type: Handshake
Limit interconnect pipeline stages to: 1

More settings can be found in the **Interconnect Requirements** tab.

System Identifier

Generation Id: 0

13. Выполните команду File=>Save и откройте окно настройки формирования описания системы: Generate=>Generate.

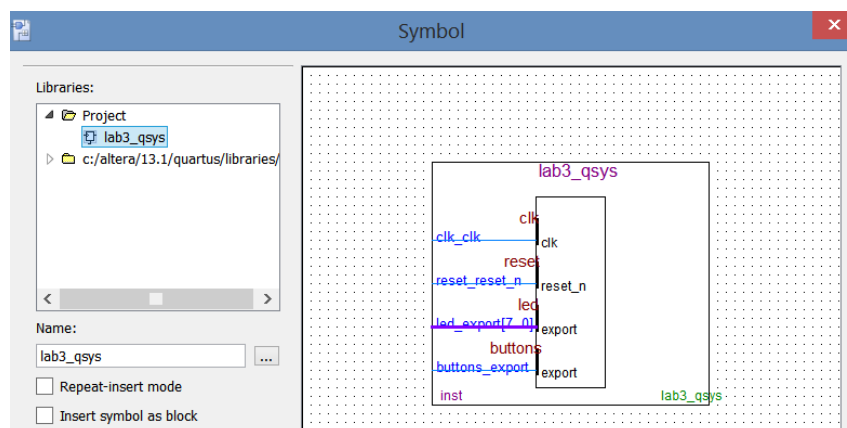


14. Оставьте все значения по умолчанию и нажмите кнопку Generate.

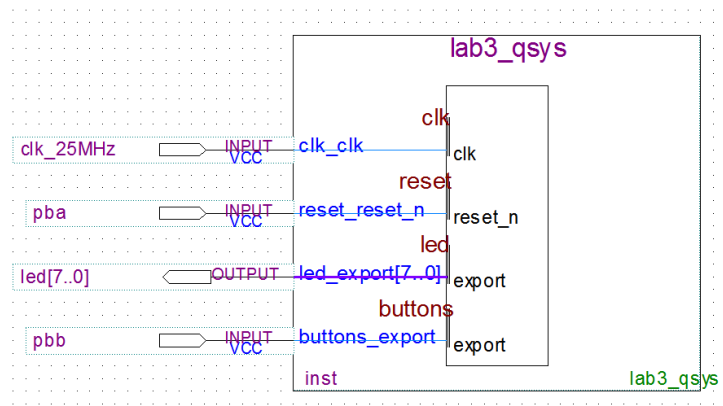
Часть 3 – Интеграция аппаратной части проекта

1. Создайте в графическом редакторе файл верхнего уровня в иерархии проекта.

- ✓ Выполните команду File=>New, укажите Block Diagram/schematic file и нажмите кнопку ОК. Откроется окно графического редактора.
- ✓ Дважды щелкните левой клавишей мыши в рабочем поле графического редактора. Откроется окно ввода символов –Symbol.
- ✓ В разделе Libraries откройте папку Project. В ней находится символ созданной системы на кристалле – lab3_qsys.



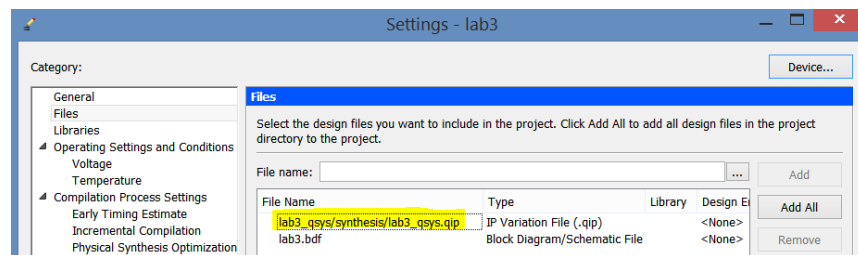
- ✓ Выберите этот символ, нажмите кнопку ОК, затем разместите его в поле графического редактора, нажав левую клавишу мыши.
- ✓ Введите схему представленную на рисунке.



- ✓ Сохраните схему под именем lab3.bdf

2. Проверка синтаксиса проекта.

- ✓ Подключите файл с описанием созданной в Qsys системы к проекту:
 - Выполните команду Project=>Add\Remove Files in Project
 - В появившемся окне, в разделе File Name выберите (с помощью браузера) файл ... \labs \lab3\lab3_qsys\synthesis
 - Нажмите кнопку Add



- Затем нажмите кнопку OK.

- ✓ Выполните команду Processing=>Start=>Start Analysis and Elaboration

3. Назначение выводов проекта.

- ✓ Запустите редактор назначения выводов (Pin Planner): Assignment=>Pin Planner.
- ✓ Назначьте выводы так, как показано на рисунке ниже

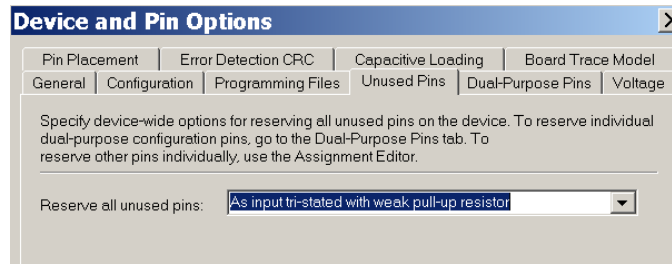
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Current Strength	Slew Rate
altera_reserved_tck	Input				2.5 V (default)	8mA (default)	
altera_reserved_tdi	Input				2.5 V (default)	8mA (default)	
altera_reserved_tdo	Output				2.5 V (default)	8mA (default)	2 (default)
altera_reserved_tms	Input				2.5 V (default)	8mA (default)	
clk_25MHz	Input	PIN_23	1	B1_N0	3.3-V LVTTTL	8mA (default)	
led[7]	Output	PIN_65	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[6]	Output	PIN_66	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[5]	Output	PIN_67	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[4]	Output	PIN_68	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[3]	Output	PIN_69	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[2]	Output	PIN_70	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[1]	Output	PIN_71	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
led[0]	Output	PIN_72	4	B4_N0	2.5 V (default)	8mA (default)	2 (default)
pba	Input	PIN_64	4	B4_N0	2.5 V (default)	8mA (default)	
pbb	Input	PIN_58	4	B4_N0	2.5 V (default)	8mA (default)	

- ✓ Закройте редактор назначения выводов.

4. Назначение опции проекта

- ✓ Выполните команду: Assignment=>Device.
- ✓ В появившемся окне нажмите кнопку Device and Pin Options

- ✓ В окне Device and Pin Options выберите закладку Unused pin, в которой установите опцию As input tri-stated with weak pull-up resistor



- ✓ Нажмите кнопку ОК. В следующем окне нажмите кнопку ОК.

Интеграция аппаратной части проекта и задание установок проекта завершено.

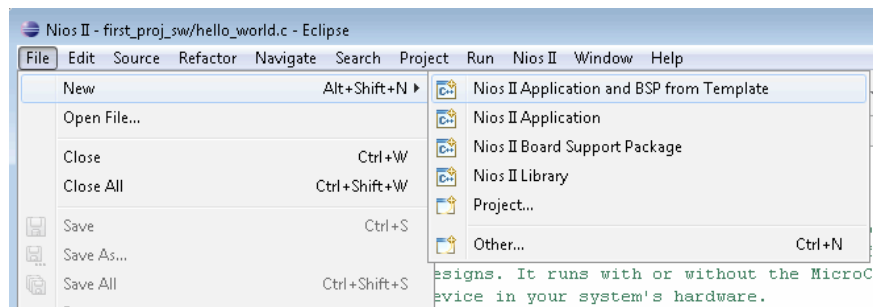
Часть 3 – Полная компиляция проекта

1. В окне менеджера пакета QuartusII, с помощью команды **Processing => Start Compilation** осуществите полную компиляцию проекта.

Часть 4 – Создание программной части проекта

1. Запустите оболочку для разработки/отладки программ - NIOSII SBT – из редактора Qsys:

- ✓ Tools=> NiosII Software Build Tools for Eclipse
- ✓ Выполните команду File=>New=>NIOS II Application and BSP from Template. Будет запущен помощник создания нового проекта – New Project Wizard



- ✓ В окне помощника введите:
 - В разделе Select Target Hardware с помощью браузера найдите в рабочей папке и укажите файл lab3_qsys.sopcinfo – файл с описанием созданной системы на кристалле.
 - В разделе Select Project Template выберите Blank Project
 - В разделе Name введите название проекта – lab3_sw
 - Нажмите кнопку Finish.
- ✓ Выполните команду File=>New=>Other. В появившемся окне выберите Source File в категории C/C++. Нажмите кнопку Next.
- ✓ В окне New source file (с помощью браузера) укажите папку lab3_sw, введите название файла: lab3_source.c; Template => Default C source template.
- ✓ Будет создан и открыт в текстовом редакторе новый файл.
- ✓ Введите текст программы на языке Си (или скопируйте из файла lab2_source.c (папка source files)):

```

// #include "sys\alt_stdio.h"
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include <unistd.h>
#include <stdio.h>
#define NONE_PRESSED 0x1    // Value read from button PIO when no buttons
pressed
#define DEBOUNCE 30000    // Time in microseconds to wait for switch
debounce

int main(void) {
    int buttons;           // Use to hold button value
    int led = 0x00;        // Use to write to led

    printf("Привет XXXXXXXX !\n Процессор Nios II запущен!\n ");
    printf("Нажмите кнопку на плате miniDiLaB-CIV\n \n ");

    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, led);    // Write new value to
pio

    while (1)
    {
        // Read buttons via pio
        buttons = IORD_ALTERA_AVALON_PIO_DATA(BUTTONS_BASE);

        if (buttons != NONE_PRESSED)    // if button pressed
        {
            if (led >= 0x80 || led==0x00)
                led = 0x01;            // reset pattern
            else
                led = led << 1;

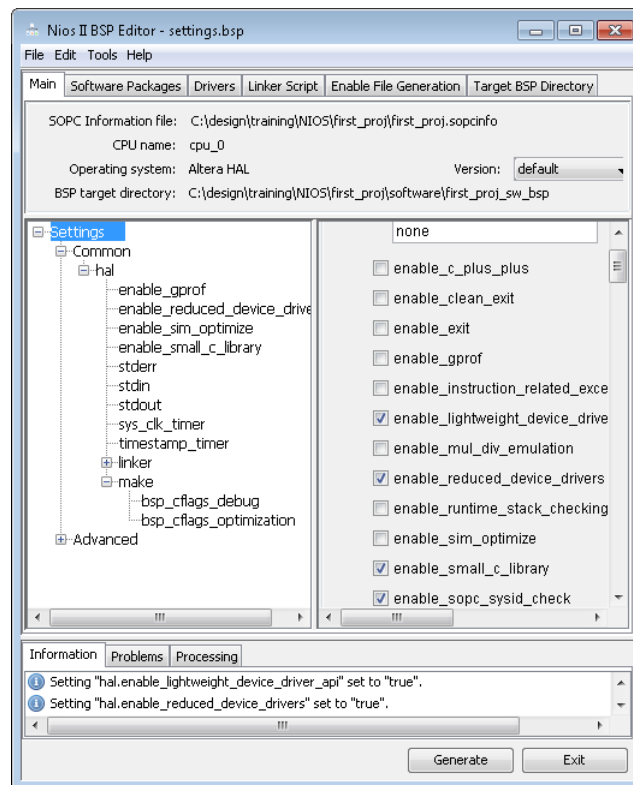
            printf("Нажата кнопка pbb\n ");

            IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, ~led);    // Write new
value to pio

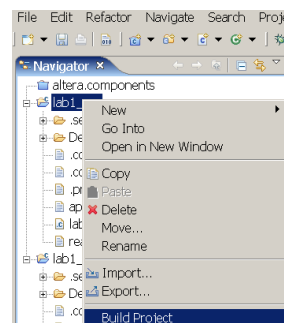
            // Switch debounce routine
            usleep (DEBOUNCE);
            while (buttons != NONE_PRESSED)    // wait for button release
                buttons = IORD_ALTERA_AVALON_PIO_DATA(BUTTONS_BASE);
            usleep (DEBOUNCE);
        }
    }
}

```

- ✓ Сохраните его.
- ✓ Выберите папку lab2_sw_bsp, нажмите правую клавишу мыши и выберите команду NIOS II => BSP Editor
- ✓ В появившемся окне на закладке Main выберите категорию Settings (все настройки) и установите опции как показано на рисунке, приведенном ниже (это поможет сократить объем порождаемого файла с исполняемым кодом программы). Нажмите кнопку Generate.



- ✓ Затем нажмите кнопку Exit.
- ✓ Выберите папку lab2_sw, нажмите правую клавишу мыши и укажите команду Build Project. Будет запущен компилятор.



Часть 5 – Конфигурирование СБИС и проверка работы на плате

1. Программирование СБИС и запуск программы в режиме отладки :

- ✓ Подсоедините USB-Blaster к **JTAG** разъему платы DiLaB

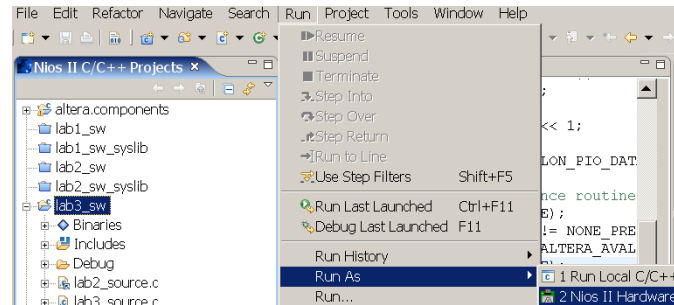


- ✓ Включите питание платы.
- ✓ В NIOSII IDE выполните команду NIOSII => **Quartus II Programmer**
- ✓ Откроется окно управления конфигурированием СБИС. При необходимости с помощью кнопки Hardware Setup установите имеющееся у Вас средство программирования СБИС. Нажмите кнопку Add File и выберите файл lab3.sof

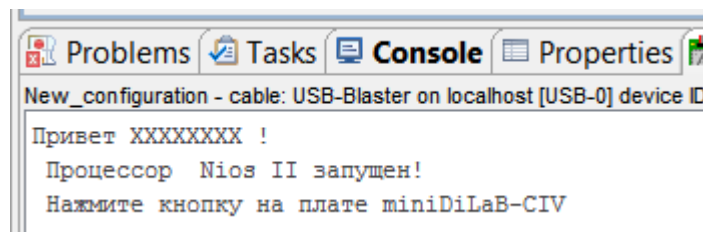
- ✓ Включите опцию **Program/Configure** и нажмите кнопку **Start**. В окне Progress будет отображаться статус процедуры программирования.

Когда СБИС будет запрограммирована на плате загорится зеленый светодиод.

2. Выберите папку lab3_sw и выполните команду Run=>Run as=>NiosII Hardware



3. Процессор будет запущен. На консоли появятся сообщения:



4. Нажмите кнопку rbb на плате miniDiLaB-CIV. Светодиод led1 включится. На консоли появится сообщение: «Нажата кнопка rbb»
5. Ожидаемый алгоритм работы программы следующий:
 - а. При каждом нажатии кнопки rbb происходит изменение номера включенного светодиода от led1 к led8 на одну позицию (с циклическим переходом от led8 к led1). Формируется сообщение «Нажата кнопка rbb» на консоли

Часть 6 – Анализ и оптимизация размера исполняемого кода программы

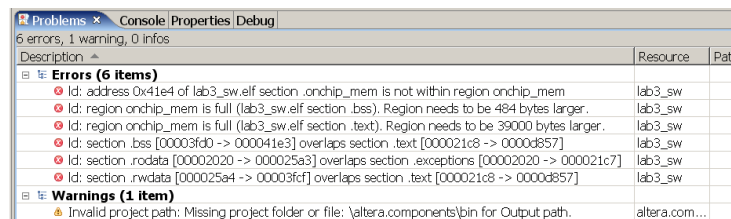
1. Для проекта *lab3_sw_bsp* установите опции BSP по умолчанию:

- Выберите проект *lab3_sw_bsp*
- нажмите правую клавишу мыши, выберите команду NIOSII => BSP Editor,
- Укажите следующие настройки:



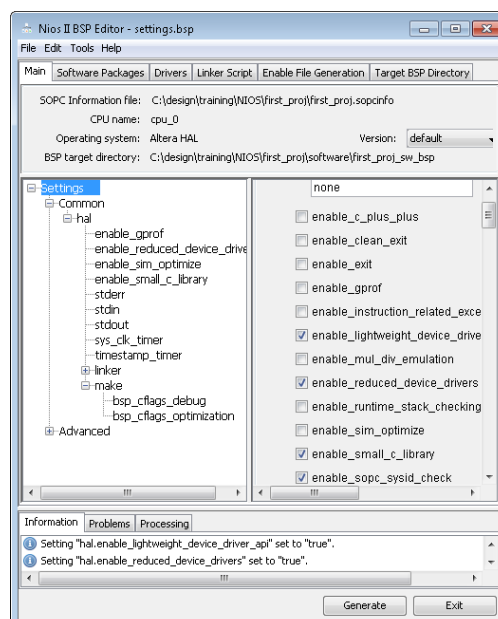
2. Для проекта *lab3_sw* выполните команду меню Project=>Build project

3. Результаты компиляции показывают, что для исходного кода программы не хватает 39000 байт (конкретное число может быть другим), т.к. размер используемой в проекте памяти для данных, программ, стека установлен 8192 байта

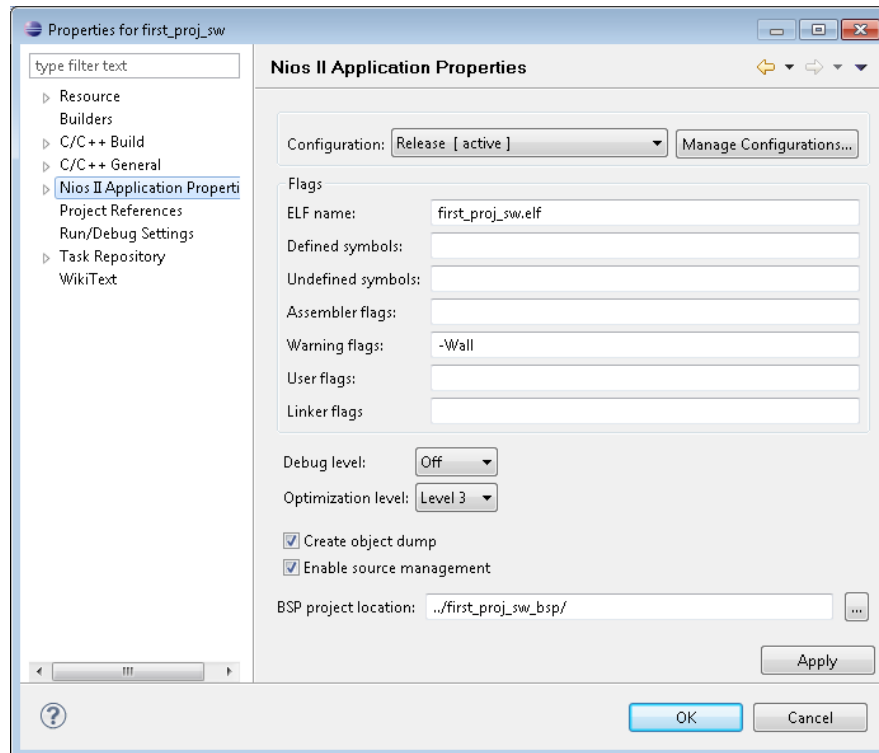


4. Верните настройки BSP, предназначенные для уменьшения размера кода

- нажмите правую клавишу мыши, выберите команду NIOSII => BSP Editor,
- Укажите следующие настройки:



5. Для проекта *lab3_sw* проверьте, что в свойствах проект разрешена опция Create objdump file:



6. Для проекта *lab3_sw* выполните команду меню Project=>Build project
7. Результаты компиляции показывают, что исходный код программы и инициализационные данные поместились в 5852 байта. При этом под стек и данные осталось 2340 байт.

```

Problems Console Properties Debug
C-Build [lab3_sw]
**** Build of configuration Debug for project lab3_sw ****
make -s all includes
Compiling lab3_source.c...
Linking lab3_sw.elf...
Info: (lab3_sw.elf) 5852 Bytes program size (code + initialized data).
Info: 2340 Bytes free for stack + heap.
Creating lab3_sw.elf.objdump...
Post-processing to create onchip_mem.hex
Hardware simulation is not enabled for the target SOPC Builder system. Skipping
hardware simulation model contents and simulation symbol files. (Note: This does
instruction set simulator.)
Build completed in 6.438 seconds

```

8. В исходном файле *lab3_source.c*:
- Подключите `alt_stdio.h`
`#include "sys\alt_stdio.h"`
 - Замените все операторы `printf()` на `alt_printf()`
9. Сохраните его под именем *lab3_source_alt.c*
10. Файл *lab3_source.c* исключите из компиляции.
11. Для проекта *lab3_sw* выполните команду меню Project=>Build project
12. Результаты компиляции показывают, что требования к объему памяти для исходного кода программ и инициализационных данных сократилось до 4260 байт. При этом под стек и данные осталось 3932 байт.

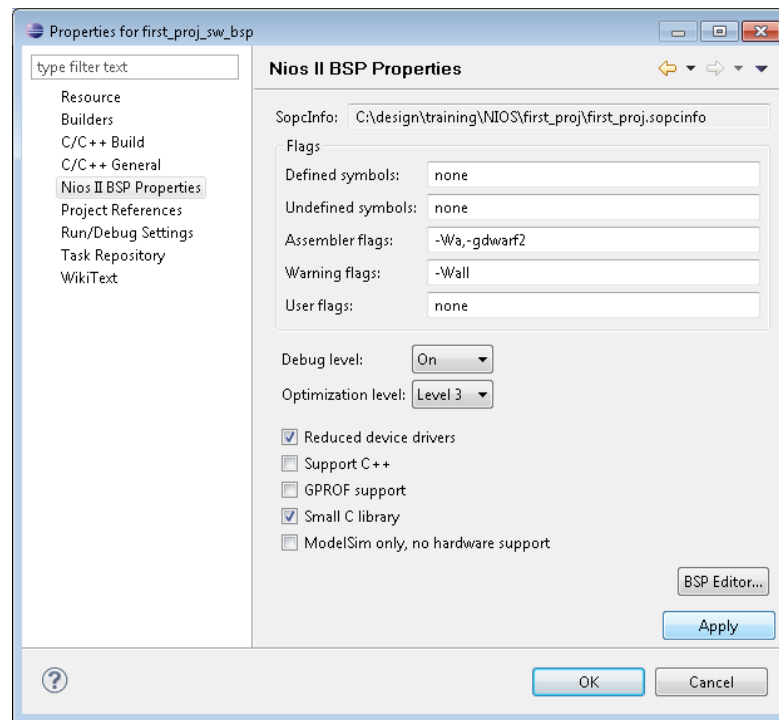
```

Problems Console Properties Debug
C-Build [lab3_sw]
**** Build of configuration Debug for project lab3_sw ****
make -s all includes
Compiling lab3_source_alt.c...
Linking lab3_sw.elf...
Info: (lab3_sw.elf) 4260 Bytes program size (code + initialized data).
Info: 3932 Bytes free for stack + heap.
Creating lab3_sw.elf.objdump...
Post-processing to create onchip_mem.hex
Hardware simulation is not enabled for the target SOPC Builder system. Skip
hardware simulation model contents and simulation symbol files. (Note: Thi
instruction set simulator.)
Build completed in 5.453 seconds

```

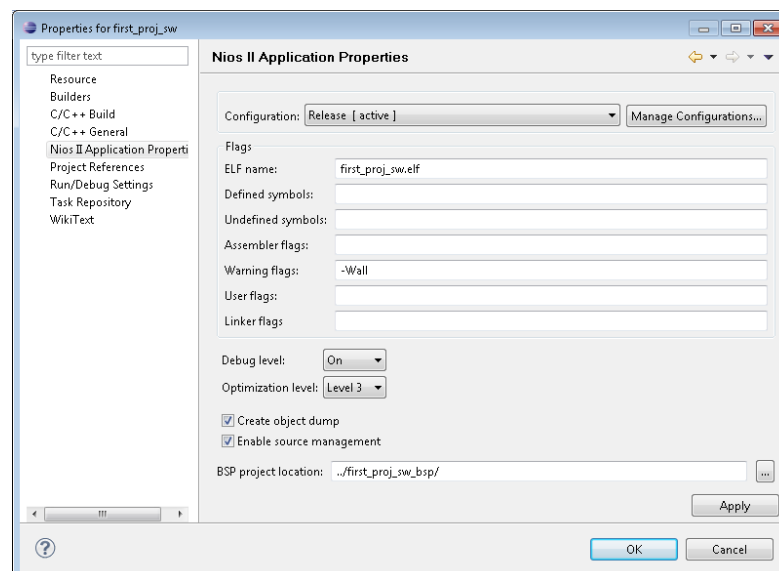
13. Выберите проект *lab3_sw_bsp*,

- нажмите правую клавишу мыши, выберите команду properties,
- переключитесь на закладку Nios II BSP Properties и установите Optimization Level равным Level 3. Нажмите кнопку OK.



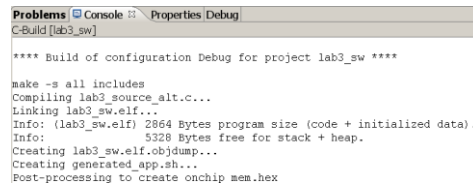
14. Выберите проект *lab3_sw*,

- нажмите правую клавишу мыши, выберите команду properties,
- переключитесь на закладку Nios II Application Properties и установите Optimization Level равным Level 3. Нажмите кнопку OK



15. Для проекта *lab3_sw* выполните команду меню Project=>Build project

16. Результаты компиляции показывают, что требования к объему памяти для исходного кода программ и инициализационных данных сократилось до 2864 байт. При этом под стек и данные осталось 5328 байт.



```
Problems Console Properties Debug
C-Build [lab3_sw]

**** Build of configuration Debug for project lab3_sw ****

make -s all includes
Compiling lab3_source_elt.c...
Linking lab3_sw.elf...
Info: (lab3_sw.elf) 2864 Bytes program size (code + initialized data).
Info: 5328 Bytes free for stack + heap.
Creating lab3_sw.elf.objdump...
Creating generated_app.sh...
Post-processing to create onchip mem.hex
```

17. С помощью команды меню Tools=> QuartusII Programmer запрограммируйте плату.
18. Для проекта lab3_sw выполните команду Run => Debug as => NiosII Hardware (запустите проект в режиме отладки) и убедитесь, что проект работает на плате DiLaB и использованные Вами ранее функции отладки доступны.
19. Выберите проект lab3_sw_bsp установите Optimization Levels равным Size.
20. Выберите проект lab3_sw, установите Optimization Levels равным Size.
21. Для проекта lab3_sw выполните команду меню Project=>Build project
22. Проанализируйте результаты компиляции.

Упражнение 3 завершено.