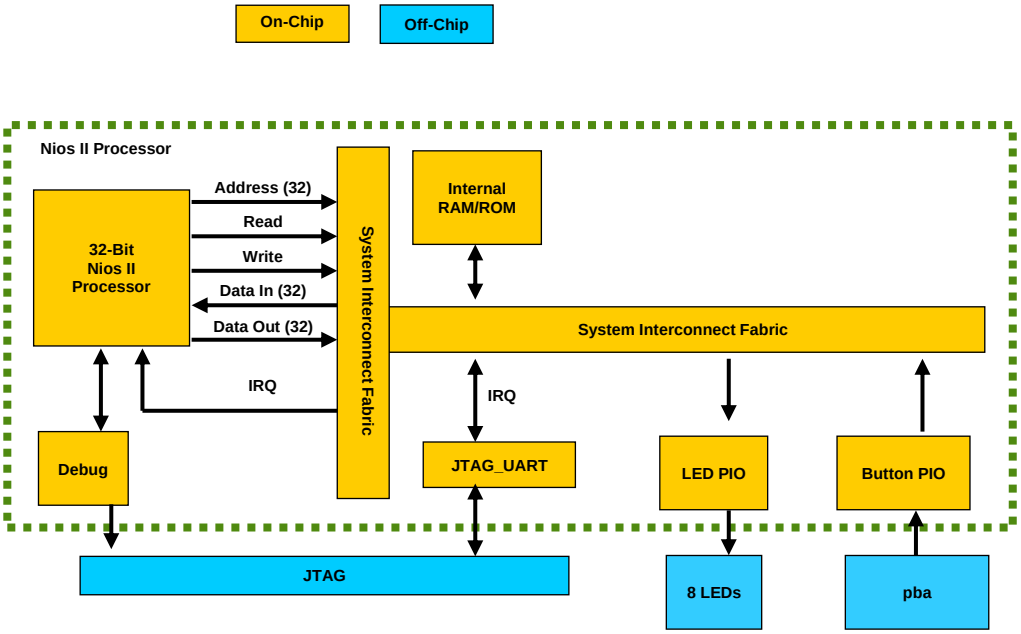
Задание labn_3

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 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)

Часть 1 – Создание проекта

1. Запустите пакет Quarus
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*)	***C:\Intel_trn\Q_NIOS*****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

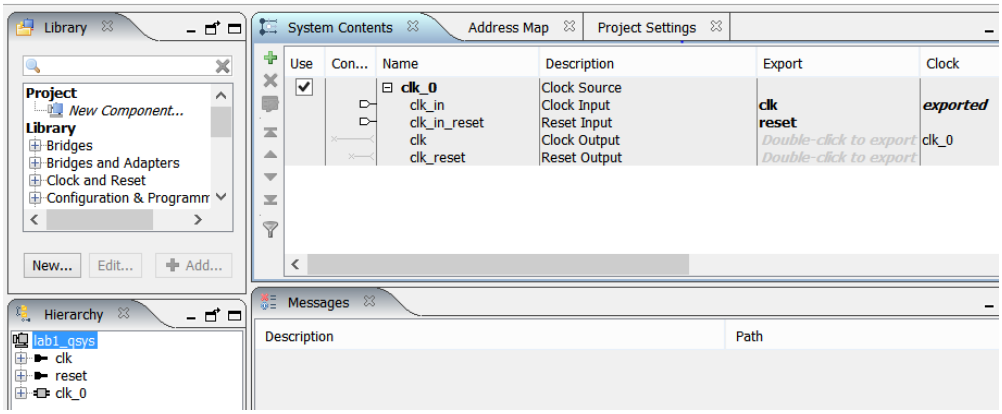
5. Нажмите кнопку **Next**.
6. В окне **Add Files [page 2 of 5]** нажмите кнопку **Next.**
7. В окне **Family & Device Setting[page3 of 5]**:
   - в разделе **Family** укажите **Cyclone IV E**.
   - в разделе **Available devices** укажите СБИС **EP4CE6E22C8**.

   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 **Lab3_nios.qsys**

3. Выберите компонент **clk_0**, нажмите правую клавишу мыши и выполните команду Edit. Задайте:

   - ✓ частоту тактового сигнала = 25 МГц, что соотствует частоте кварцевого генератора на плате miniDiLaB-CIV.
   - ✓ Параметр Reset synchronous edges => Deassert
   - ✓ Нажмите кнопку Finish.

4. Переименуйте компонент **clk_0**: выберите компонент **clk_0**, нажмите правую клавишу мыши и выберите команду **Rename**. Новое имя - **clk**.

5. Создание, на основе встроенных модулей М9К, 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 (Writable)
     - В разделе Total Memory Size задайте размер памяти: 16384 байт.
     - В разделе memory Initialization включите опцию Enable non-default initialization file (имя файла д.б. onchip_mem.hex)
     - Нажмите кнопку Finish.

Так как, для нормальной работы созданны 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.

6. Переименуйте созданный модуль памяти **onchip_memory2_0**: выберите компонент **onchip_memory2_0**, нажмите правую клавишу мыши и выберите команду **Rename**. Новое имя - **onchip_mem**.

7. Соедините выход **clk** компонента **clk** с входом **clk1** компонента **onchip_mem**, а выход **clk_reset** компонента **clk** с входом **reset1** компонента **onchip_mem**.

8. Конфигурация ядра процессора NIOSII

   - ✓ В списке доступных компонентов выберите раздел Processors => NIOSII Processor и дважды щелкните левой клавишей мыши. Появится окно конфигурации процессора.
     - ✓ На закладке Main выберите:
       - тип процессора - NIOSII/**e** –(простейший вариант процессорного ядра)
     - ✓ Переключитесь на закладку JTAG Debug и установите параметр Include JTAG Debug.

- ✓ Нажмите кнопку Finish.

Появляющиеся ошибки и предупреждения пока можно проигнорировать.

9. Переименуйте созданный процессорный модуль **nios2_gen2_0**: выберите компонент **nios2_gen2_0**, нажмите правую клавишу мыши и выберите команду **Rename**. Новое имя - **nios2_qsys**.

10. Соедините вход **clk** компонента **nios2_qsys** с выходом **clk** компонента **clk**, а выход **clk_reset** компонента **clk** с входом reset компонента **nios2_qsys**.

11. Соедините вход s1 компонента onchip_mem с выходами data_master и instruction_master компонента nios2_qsys.

12. Откройте окно настройки процессора Nios II: выберите компонент **nios2_qsys**, нажмите правую клавишу мыши и выполните команду Edit

    - ✓ На закладке Vectors укажите
      - память для вектора сброса - onchip_mem.s1
      - память для вектора exception - onchip_mem.s1
    - ✓ Нажмите кнопку Finish.

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

    - ✓ В списке доступных компонентов выберите модуль 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.

14. Переименуйте созданный компонент pio_0: выберите компонент pio_0, нажмите правую клавишу мыши и выберите команду Rename. Новое имя - **led**.

15. Соедините вход clk компонента led с выходом clk компонента clk, а выход clk_reset компонента clk с входом reset компонента led.

16. Соедините вход s1 компонента led с выходом data_master компонента nios2_qsys.

Появляющиеся ошибки и предупреждения пока можно проигнорировать.

17. Экспортируйте вывод компонента led: дважды щелкните в поле строки external_connection, столбца Export и введите имя внешнего вывода - led

18. Конфигурация и подключение к системе еще одного модуля PIO.

    - ✓ 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.

19. Переименуйте созданный компонент pio_0: выберите компонент pio_0, нажмите правую клавишу мыши и выберите команду Rename. Новое имя - **buttons**.

20. Соедините вход clk компонента buttons с выходом clk компонента clk, а выход clk_reset компонента clk с входом reset компонента buttons.

21. Соедините вход s1 компонента buttons с выходом data_master компонента nios2_qsys.

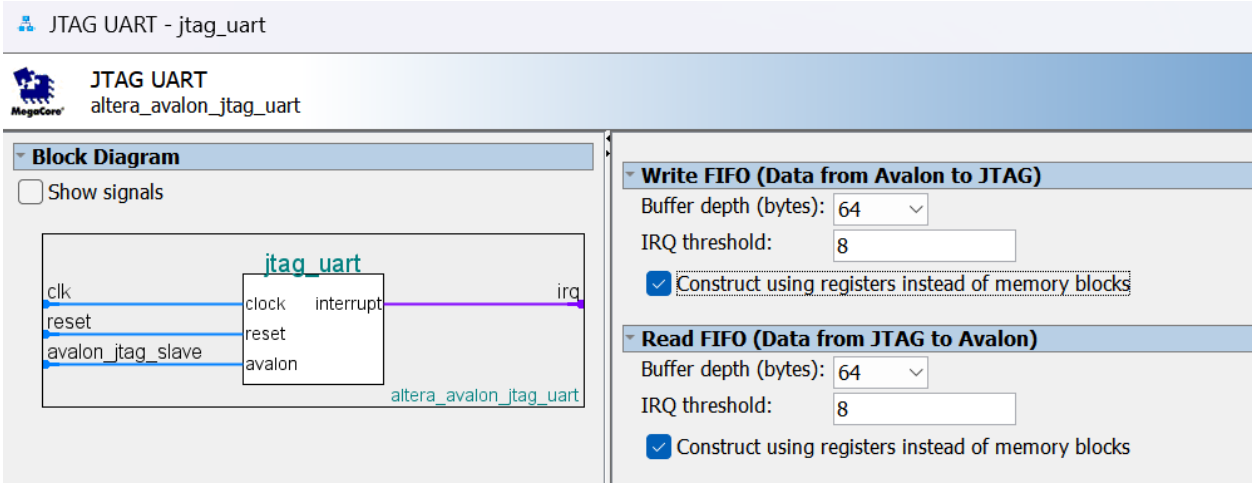
Появляющиеся ошибки и предупреждения пока можно проигнорировать.

22. Экспортируйте вывод компонента buttons: дважды щелкните в поле строки external_connection, столбца Export и введите имя внешнего вывода - pba

23. Конфигурация и подключение к системе модуля JTAG UART, используемого для передачи символьных данных между процессором и ПК через интерфейс JTAG (USB Blaster…). Будет использован как порт для стандартного ввода вывода процессора (операторов print()).

    ✓ В списке доступных компонентов выберите раздел JTAG UART и дважды щелкните левой клавишей мыши. Откроется окно настройки.

    - Установите в нем значения как показано на рисунке.



    - Нажмите кнопку Finish.

Появляющиеся ошибки и предупреждения пока можно проигнорировать.

24. Переименуйте созданный компонент jtag_uart_0: выберите компонент jtag_uart_0, нажмите правую клавишу мыши и выберите команду Rename. Новое имя - **jtag_uart**.

25. Соедините вход clk компонента jtag_uart с выходом clk компонента clk, а выход clk_reset компонента clk с входом reset компонента jtag_uart.

26. Соедините вход avalon_jtag_slave компонента jtag_uart с выходом data_master компонента nios2_qsys.

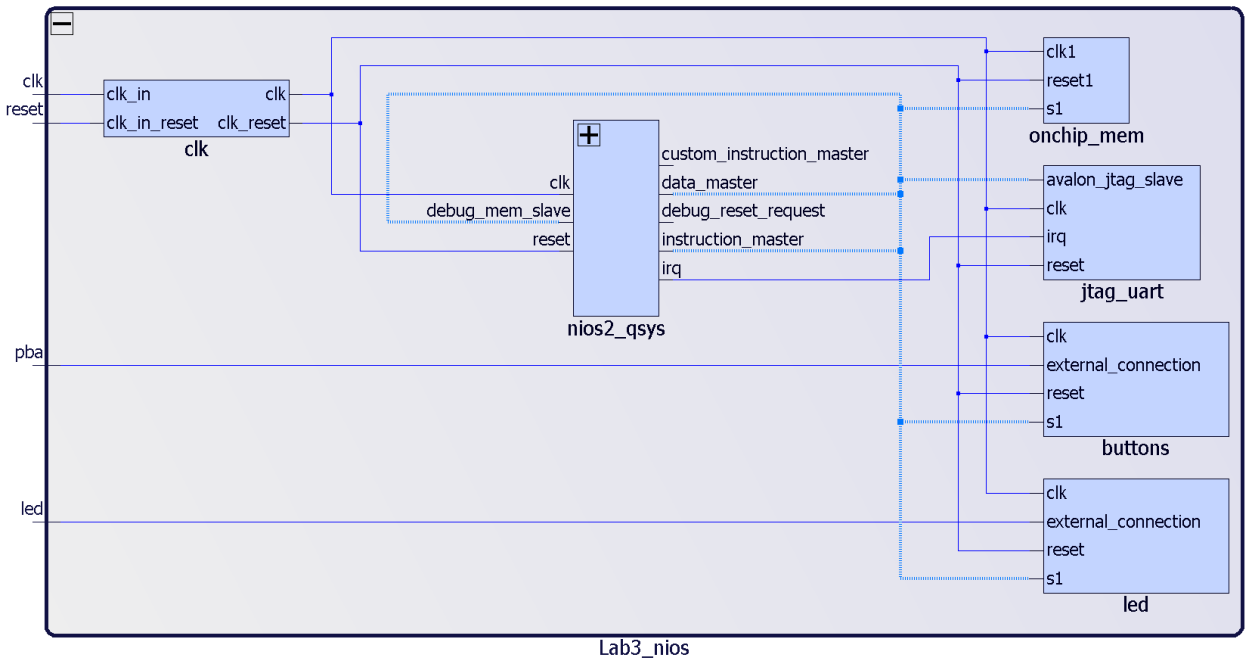
27. Соедините выход прерывания irq компонента jtag_uart с входом прерывания irq компонента nios2_qsys. *Обратите внимание на колонку IRQ.*

28. Выполните автоматическое распределение адресного пространства системы: System=>Assign base Addresses

29. Внешний вид созданной Вами системы, закладка System Contents, должен примерно соответствать приведенному на рисунке.

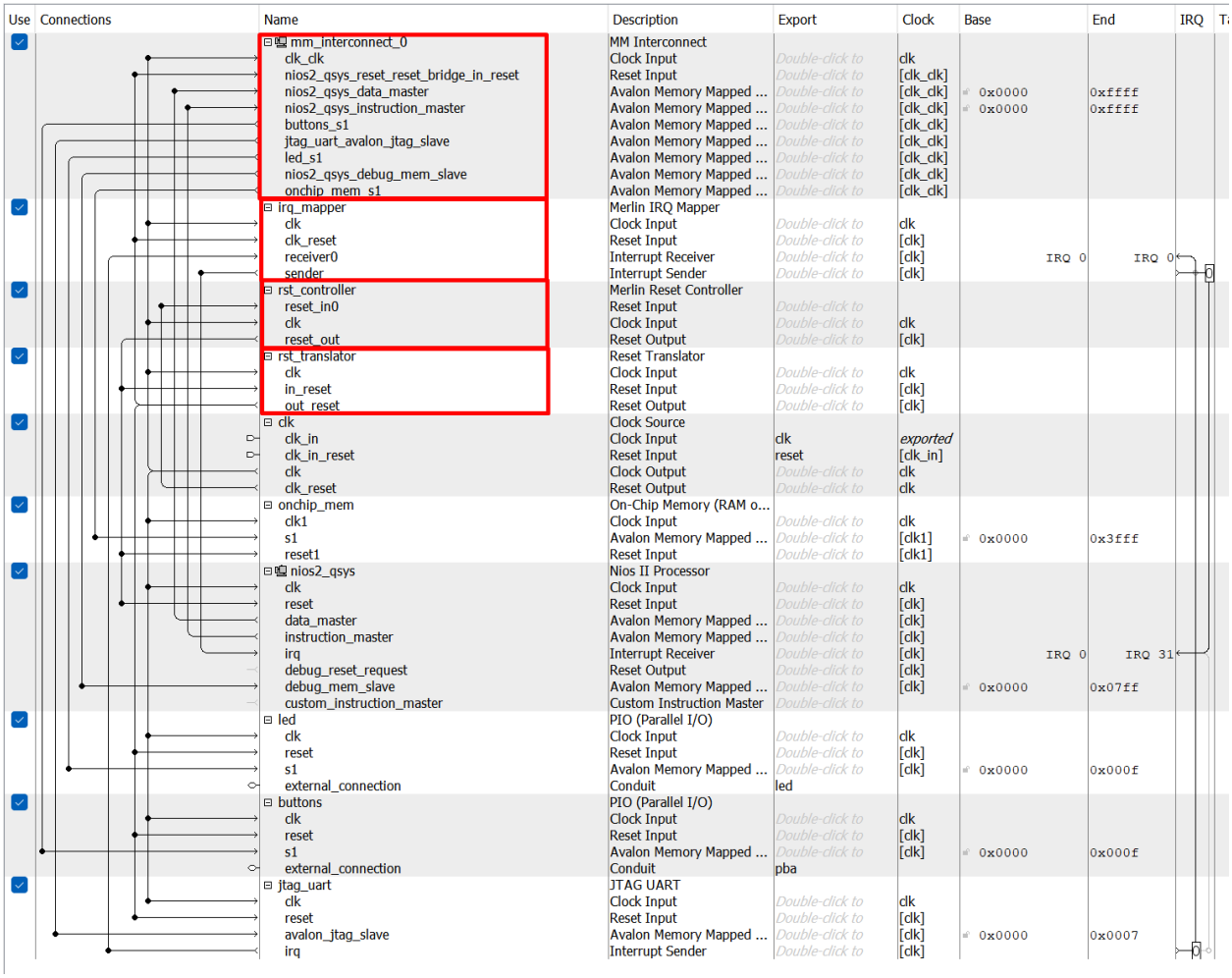
Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
		clk	Clock Source					
		clk_in	Clock Input	clk	exported			
		clk_in_reset	Reset Input	reset	[clk_in]			
		clk	Clock Output	Double-click to export	clk			
		clk_reset	Reset Output	Double-click to export	clk			
✓		onchip_mem	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	Double-click to export	clk			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x4000	0x7fff	
		reset1	Reset Input	Double-click to export	[clk1]			
✓		nios2_qsys	Nios II Processor					
		clk	Clock Input	Double-click to export	clk			
		reset	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		irq	Interrupt Receiver	Double-click to export	[clk]	IRQ 0	IRQ 31	
		debug_reset_request	Reset Output	Double-click to export	[clk]			
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x8800	0x8fff	
		custom_instruction_m...	Custom Instruction Master	Double-click to export				
✓		led	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9010	0x901f	
		external_connection	Conduit	led				
✓		buttons	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9000	0x900f	
		external_connection	Conduit	pba				
✓		jtag_uart	JTAG UART					
		clk	Clock Input	Double-click to export	clk			
		reset	Reset Input	Double-click to export	[clk]			
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9020	0x9027	
		irq	Interrupt Sender	Double-click to export	[clk]			

30. Проверьте схемное отображение созданной системы: View=>Schematic



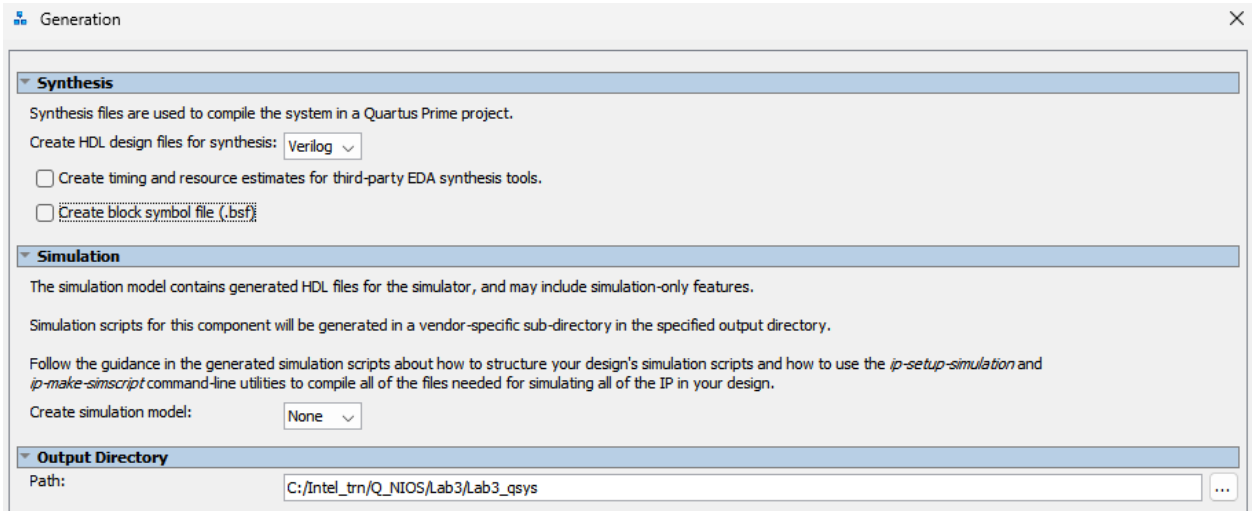
31. Выполните команду System=> Show System with Qsys interconnect, проверьте полученную структуру проекта

✓ Обратите внимание на модули, которые были добавлены в систему автоматически



32. Выполните команду File=>Save

33. Выполните команду Generate=>Generate HDL.



34. Нажмите кнопку Generate.

35. Не должно появится ошибок генерации. Предупреждение

⚠ Warning: **Lab3_qsys.jtag_uart**: JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board

можно проигнорировать.

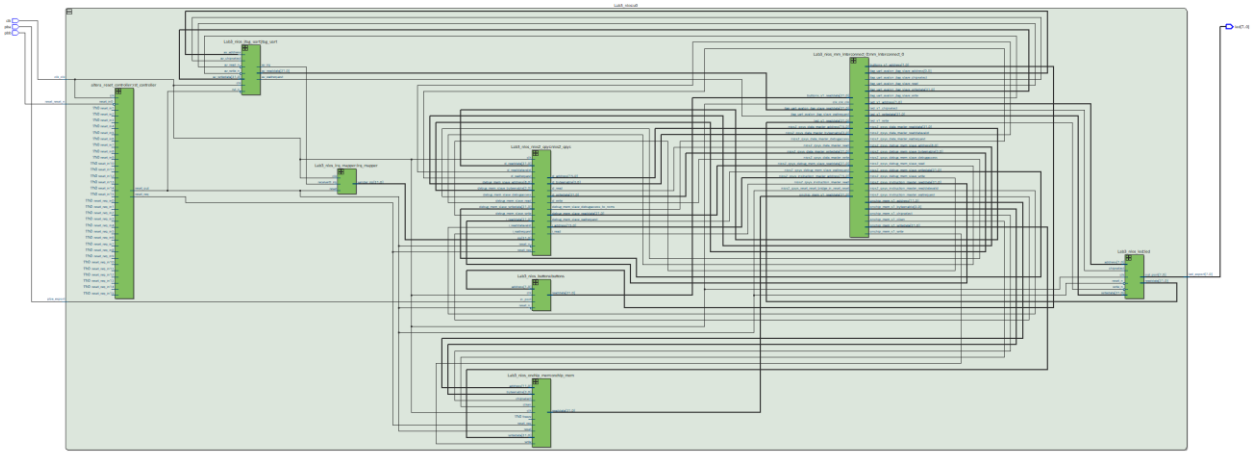
36. Нажмите кнопку Close, а затем Finish.

Часть 3 – Интеграция аппаратной части проекта

1. Создайте в текстовом редакторе файл (имя файла Lab3.sv) верхнего уровня в иерархии проекта (для этого целесообразно использовать файл Lab3_nios_inst.v из папки C:\Intel_trn\Q_NIOS\Lab3\Lab3_nios)
    a. Пример файла приведен на рисунке

```
1  module Lab3 (
2      input bit clk,       // Clock
3      input bit pbb,       // System reset active low
4      input bit pba,       // button
5      output bit [7:0] led
6  );
7  Lab3_nios u0 (
8      .clk_clk       (clk), //   clk.clk
9      .reset_reset_n (pbb), // reset.reset_n
10     .led_export    (led), //   led.export
11     .pba_export    (pba)  //   pba.export
12 );
13 endmodule
```

2. Назначьте файл Lab3.sv файлом верхнего уровня в иерархии проекта.
3. Добавьте к проекту файл Lab3_nios.qip из папки C:\Intel_trn\Q_NIOS\Lab3\Lab3_nios\synthesis
4. Проверка синтаксиса проекта.
    ✓ Выполните команду Processing=>Start=>Start Analysis and Elaboration
5. С помощью RTL Viewer проверьте полученную структуру проекта
    a. Сравните со структурой отображаемой в Qsys



6. Назначение выводов проекта.
    ✓ Запустите редактор назначения выводов (Pin Planner): Assignment=>Pin Planner.
    ✓ Назначьте выводы как показано на рисунке ниже

Node Name	Direction	Location	I/O Bank	/REF Group	I/O Standard
clk	Input	PIN_23	1	B1_N0	3.3-V LVTTL
led[7]	Output	PIN_65	4	B4_N0	2.5 V (default)
led[6]	Output	PIN_66	4	B4_N0	2.5 V (default)
led[5]	Output	PIN_67	4	B4_N0	2.5 V (default)
led[4]	Output	PIN_68	4	B4_N0	2.5 V (default)
led[3]	Output	PIN_69	4	B4_N0	2.5 V (default)
led[2]	Output	PIN_70	4	B4_N0	2.5 V (default)
led[1]	Output	PIN_71	4	B4_N0	2.5 V (default)
led[0]	Output	PIN_72	4	B4_N0	2.5 V (default)
pba	Input	PIN_64	4	B4_N0	2.5 V (default)
pbb	Input	PIN_58	4	B4_N0	2.5 V (default)

- ✓ Закройте редактор назначения выводов.

7. 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 OK. В следующем окне нажмите кнопку OK.
8. С помощью Timing Analyzer или в текстовом редакторе создайте файл (**Lab3.sdc**) с требованиями к временным параметрам проекта. Пример файла приведен на рисунке

```
##
## DEVICE  "EP4CE6E22C8"
##


#**************************************************************
# Time Information
#**************************************************************

set_time_format -unit ns -decimal_places 3

#**************************************************************
# Create Clock
#**************************************************************

create_clock -name {clock} -period 40.000 -waveform { 0.000 20.000 } [get_ports {clk}]

#**************************************************************
# Set Clock Uncertainty
#**************************************************************

derive_clock_uncertainty

#**************************************************************
# Set Input Delay
#**************************************************************

set_input_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {pbb}]
set_input_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {pba}]

set_input_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {altera_reserved_tdi}]
set_input_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {altera_reserved_tms}]

#**************************************************************
# Set Output Delay
#**************************************************************

set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[0]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[1]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[2]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[3]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[4]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[5]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[6]}]
set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {led[7]}]

set_output_delay -add_delay  -clock [get_clocks {clock}]  10.000 [get_ports {altera_reserved_tdo}]
```

9. Подключите файл Lab3.sdc к проекту.

10. Осуществите полную компиляцию проекта: команда **Processing => Start Compilation**.
11. Компиляция должна завершиться без ошибок. Все требования к временным параметрам должны быть выполнены.

Часть 4 – Создание программной части проекта

1. Запустите оболочку для разработки/отладки программ - NIOSII SBT – из редактора Qsys: команда Tools=> NiosII Software Build Tools for Eclipse

2. Укажите рабочую папку C:\Intel_trn\Q_NIOS\Lab3\workspace

3. Выполните команду File=>New=>NIOS II Application and BSP from Tempate. Будет запущен помощник создания нового проекта. В окне помощника введите:

   - ✓ В разделе **SOPC Information File name** с помощью браузера найдите в рабочей папке проекта (C:\Intel_trn\Q_NIOS\Lab3) и укажите файл Lab3_nios.sopcinfo – файл с описанием созданной системы на кристалле.

- Убедитесь что в поле CPU name появилось правильное имя процессора – nios2_qsys

✓ В разделе Project name введите название проекта: lab3_sw

✓ В разделе Project Template выберите Blank Project

✓ Нажмите кнопку Finish.

4. Выполните команду File=>New=>Other.

5. В появившемся окне в категории C/C++ выберите Source File. Нажмите кнопку Next.

37. В окне New source file

✓ укажите папку lab3_sw,

✓ введите название файла: lab3_source.c;

✓ в поле Template задайте Default C source template.

✓ Нажмите Finish.

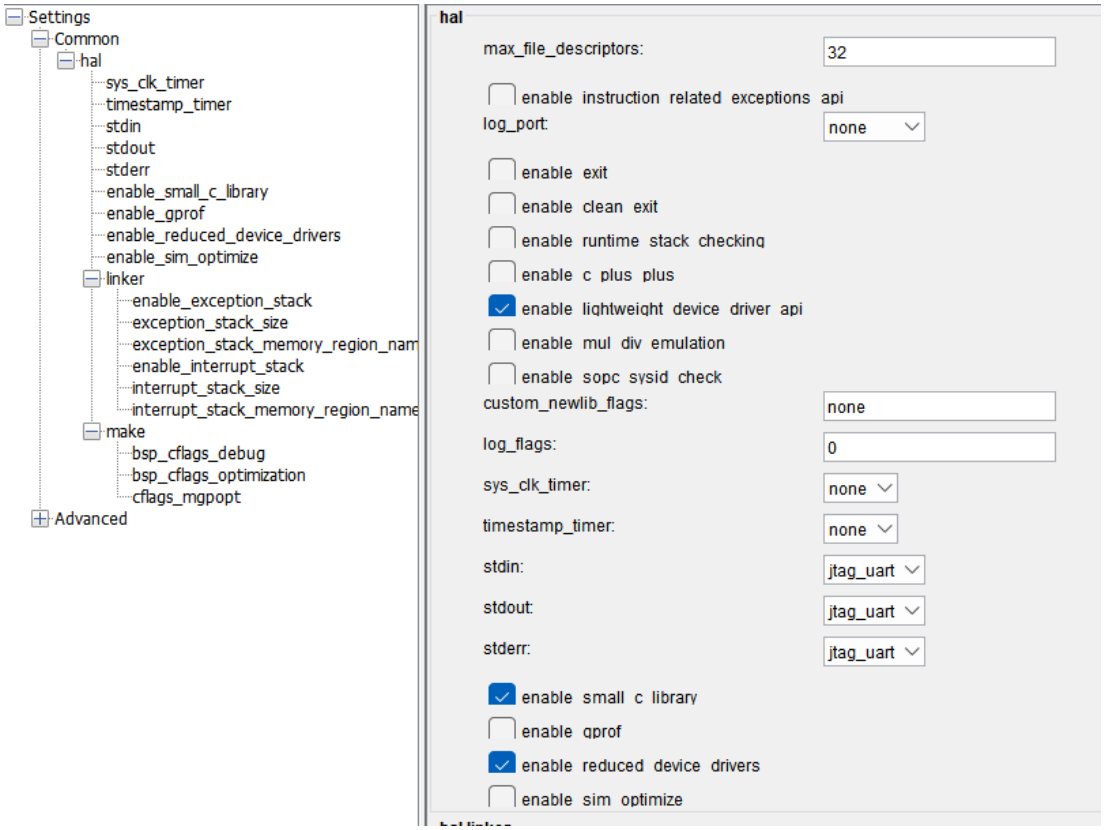
Будет создан и открыт в текстовом редакторе новый файл - lab3_source.c.

38. Введите текст программы на языке Си:

lab3_source.c

```
//#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 50000     // 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 \n Процессор  Nios II запущен!\n ");
    printf("Нажмите кнопку pba 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 pba\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);
        }
     }
 }
```

39. Сохраните его.
40. Выберите папку lab3_sw_bsp, нажмите правую клавишу мыши и выберите команду NIOS II => BSP Editor
41. В появившемся окне на закладке Main
    - ✓ выберите категорию Settings (все настройки)
    - ✓ установите опции как показано на рисунке, приведенном ниже (это поможет сократить объем порождаемого файла с исполняемым кодом программы).



    - ✓ Затем нажмите кнопку Exit, далее Yes, Save
42. Выберите папку lab3_sw_bsp, нажмите правую клавишу мыши и выполните команду NIOS II => Generate BSP
43. Выберите папку lab3_sw_bsp, нажмите правую клавишу мыши и выполните команду Build Project.
44. Выберите папку **lab3_sw**, нажмите правую клавишу мыши и выполните команду Build Project.
    - ✓ При успешном завершении процесса, в окне Console появится сообщение

```
Info: (lab3_sw.elf) 4472 Bytes program size (code + initialized data).
Info:               11 KBytes free for stack + heap.
Info: Creating lab3_sw.objdump
nios2-elf-objdump --disassemble --syms --all-header --source lab3_sw.elf >lab3_sw.objdump
[lab3_sw build complete]
```

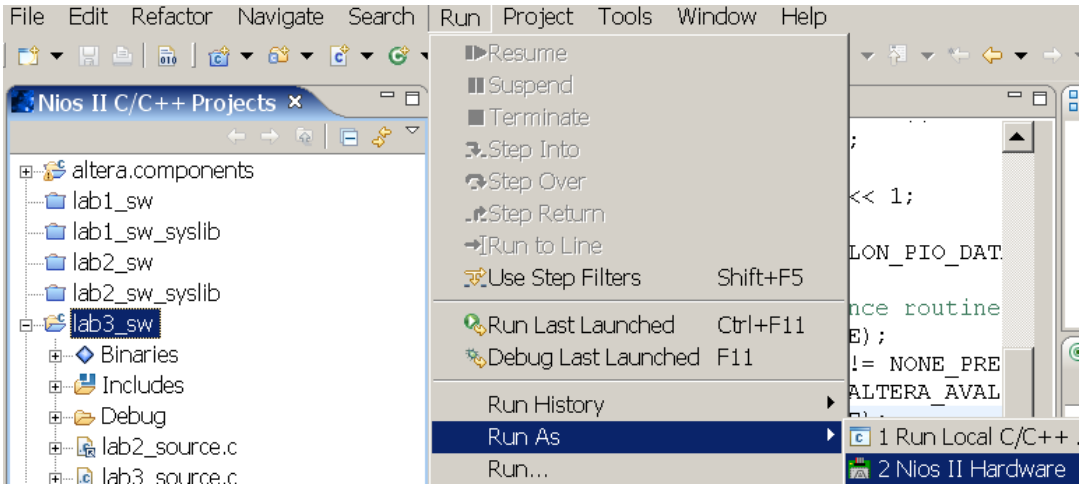
Часть 5 – Конфигурирование СБИС и проверка работы на плате

1. Программирование СБИС и запуск программы:
    - ✓ Подсоедините USB-Blaster к **JTAG** разъему платы miniDiLab
    - ✓ Включите питание платы.

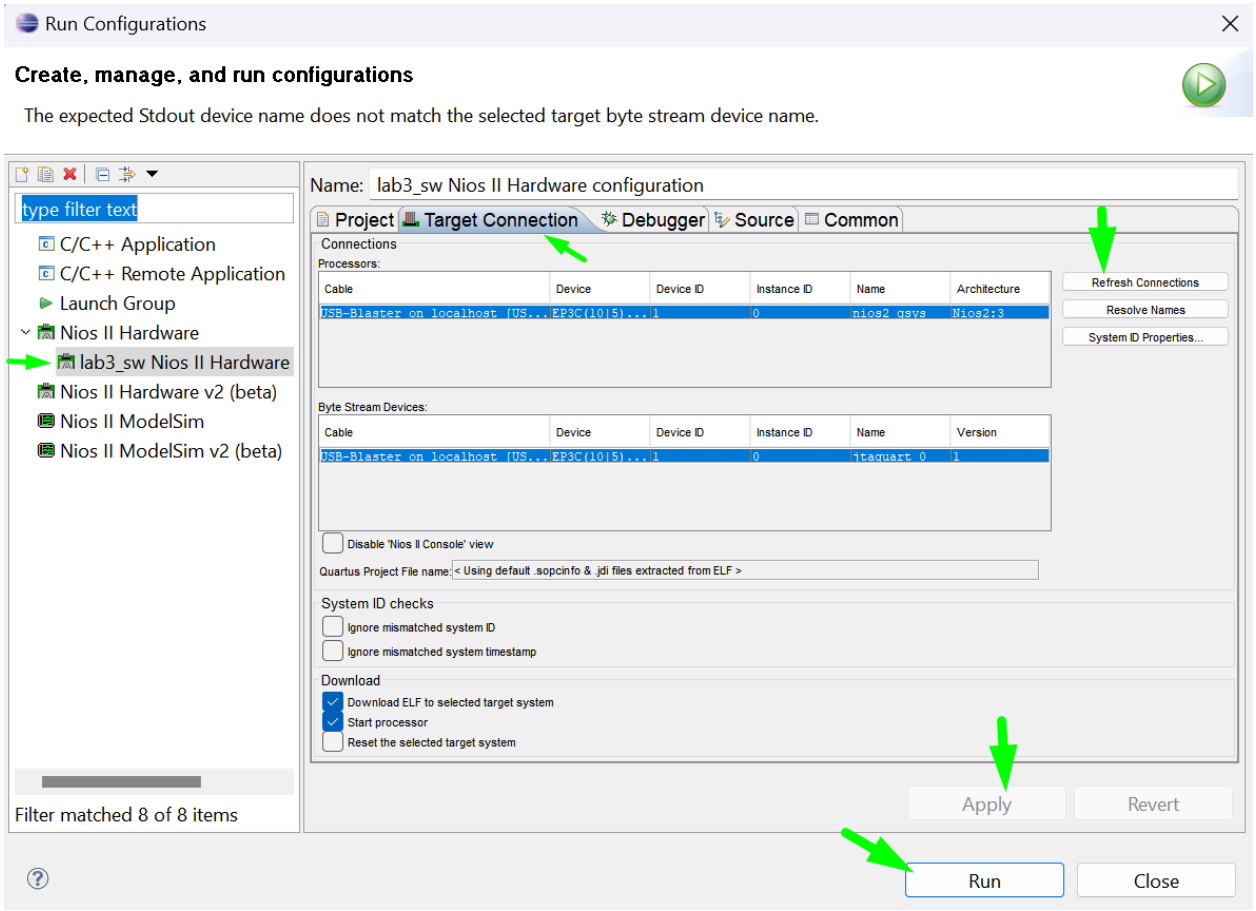
- ✓ В NIOSII IDE выполните команду **NIOSII => Quartus II Programmer**
- ✓ Откроется окно управления конфигурированием СБИС. При необходимости с помощью кнопку Hardware Setup установите имеющееся у Вас средство программирования СБИС. Нажмите кнопку Add File и выберите файл lab3.sof
- ✓ Включите опцию **Program/Configure** и нажмите кнопку **Start.** В окне Progress будет отображаться статус процедуры программирования.

Когда СБИС будет запрограммирована на платезагорится зеленый светодиод.

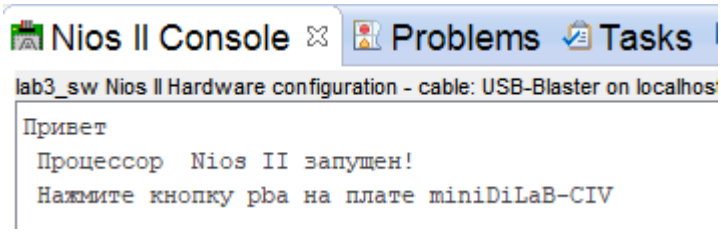
2. Выберите папку lab3_sw и выполните команду Run=>Run as=>NiosII Hardware



3. ПРИ НЕОБХОДИМОСТИ – сконфигурируйте JTAG соединение и нажмите кнопку Apply, а затем кнопку RUN



4. Процессор будет запущен. На консоли появятся сообщения:



5. Нажмите кнопку pba на плате miniDiLaB-CIV. Светодиод led1 включится. В окне консоли появится сообщение: «`Нажата кнопка pba`»

6. Ожидаемый алгоритм работы программы следующий:

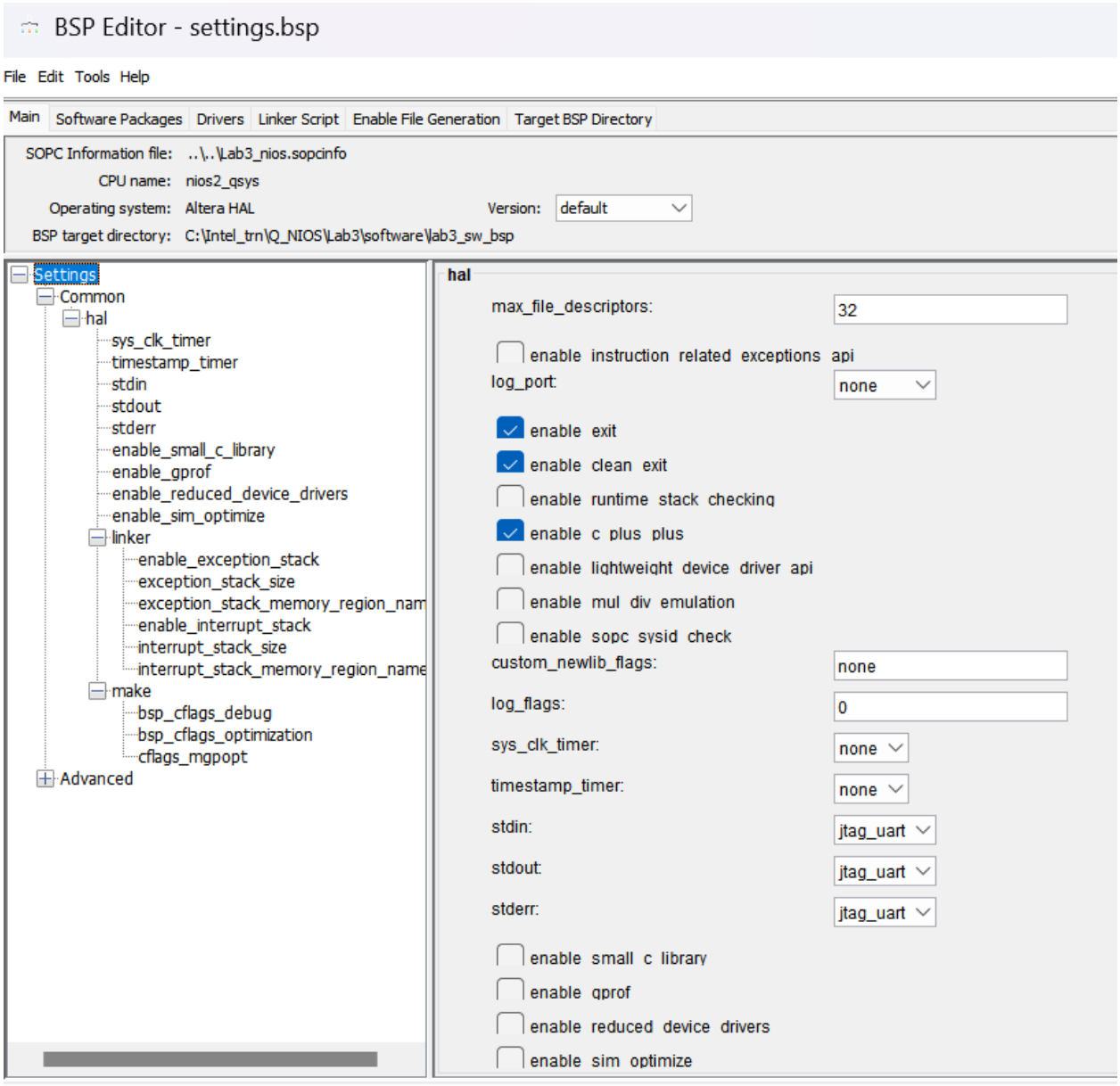
    a. При каждом нажатии кнопки pbb:

        i. происходит изменение номера включенного светодиода от led1 к led8 на одну позицию (с циклическим переходом от led8 к led1).

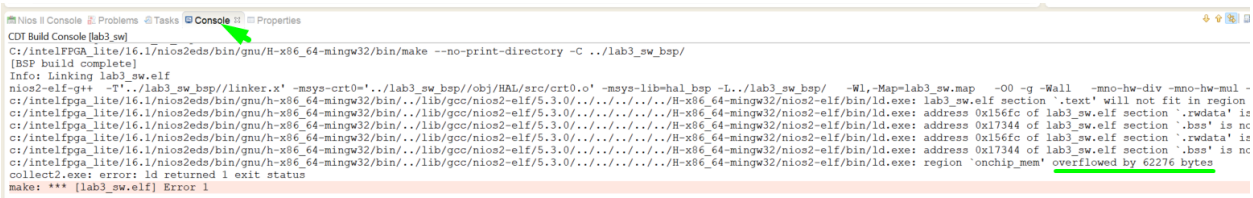
        ii. В окне консоли формируется сообщение «`Нажата кнопка pba`»

Часть 6 – Анализ и оптимизация размера исполняемого кода программы

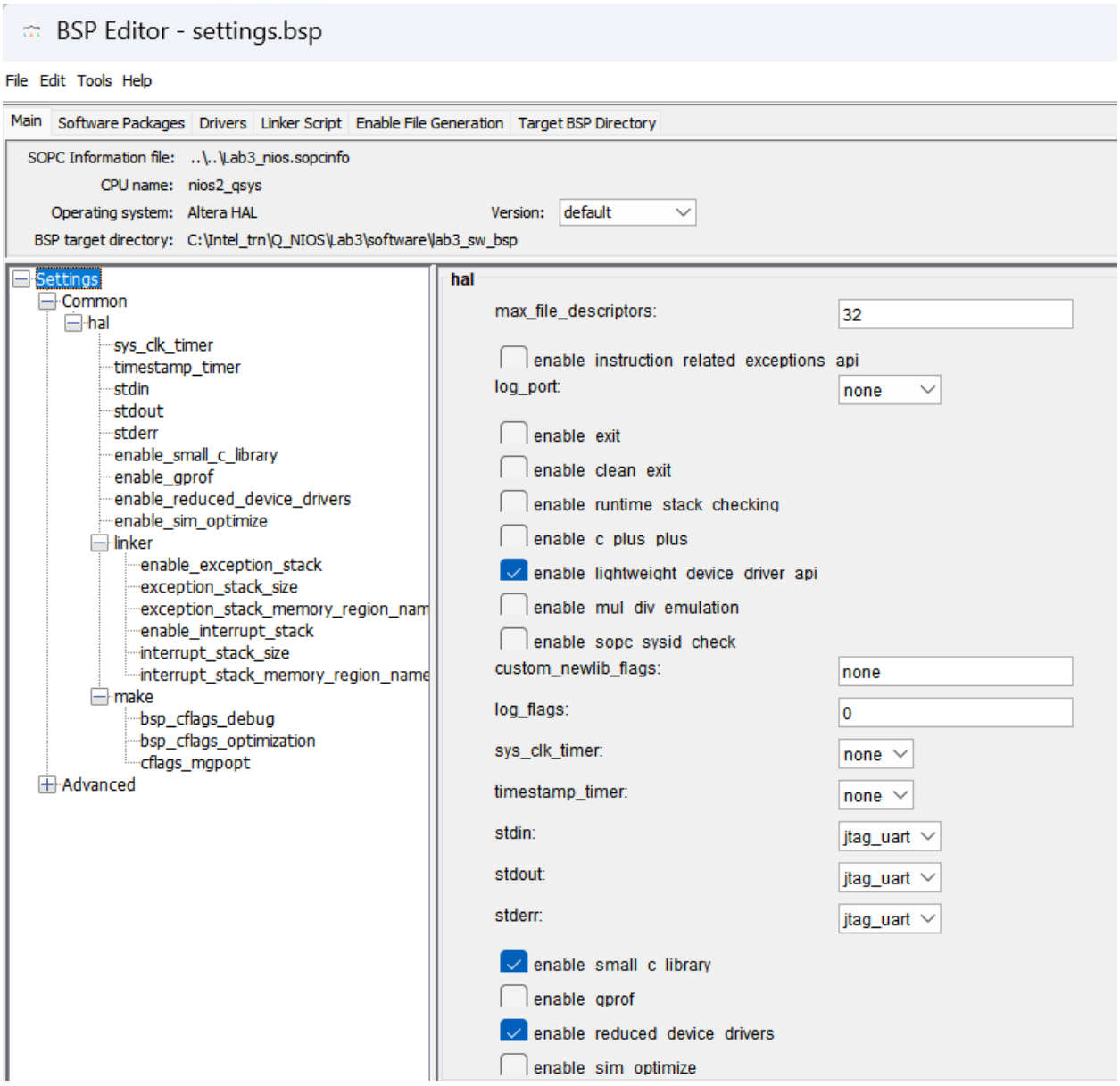
1. Для проекта *lab3_sw_bsp* установите опции BSP:
   - Выберите проект *lab3_sw_bsp*
   - нажмите правую клавишу мыши, выберите команду NIOSII => BSP Editor,
   - Укажите следующие настройки:



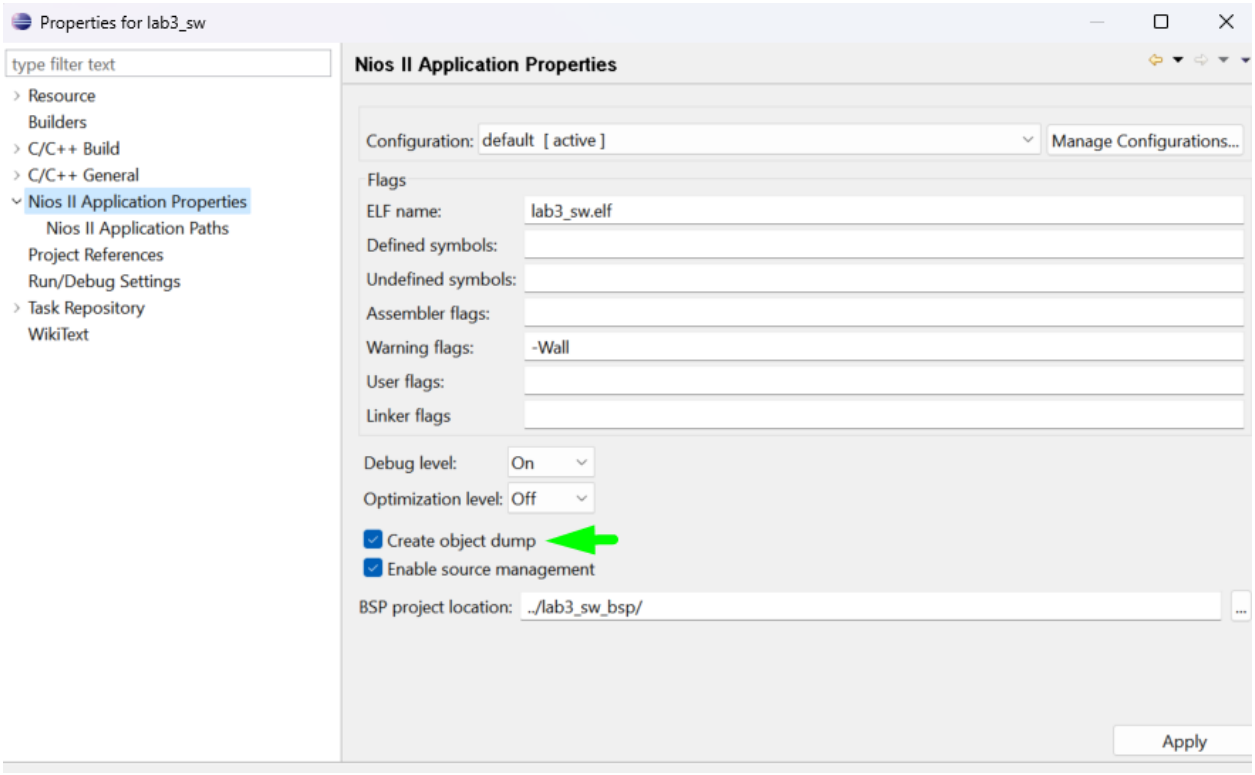
2. Нажмите кнопку Generate
3. Нажмите кнопку Exit
4. Выберите проект *lab3_sw_bsp* выполните команду меню Project=>Build project
5. Выберите проект *lab3_sw* выполните команду меню Project=>Build project
6. Результаты компиляции показывают, что для исходного кода программы не хватает 62276 байт (конкретное число может быть другим)



7. Верните настройки BSP, предназначенные для уменьшения размера кода (*что бы изменения были применены необходимо: перегенерировать BSP и выполнить Build для проекта lab3_sw_bsp)*

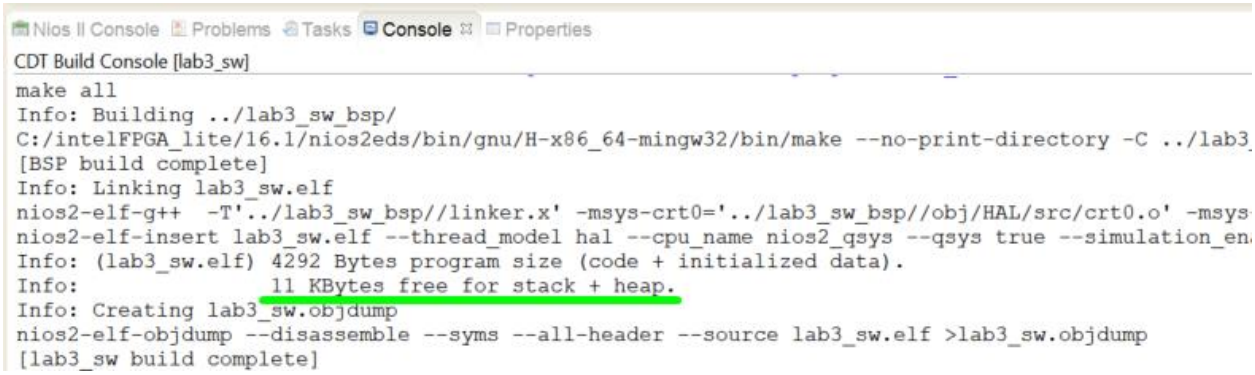


8. Для проекта *lab3_sw* проверьте, что в свойствах (Properties) проект разрешена опция Create objdump file:



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

10. Результаты компиляции показывают, что исходный код программы и инициализационные данные поместились в 4292 байта. При этом под стек и данные осталось 11К байт.



11. В исходном файле lab3_source.c:

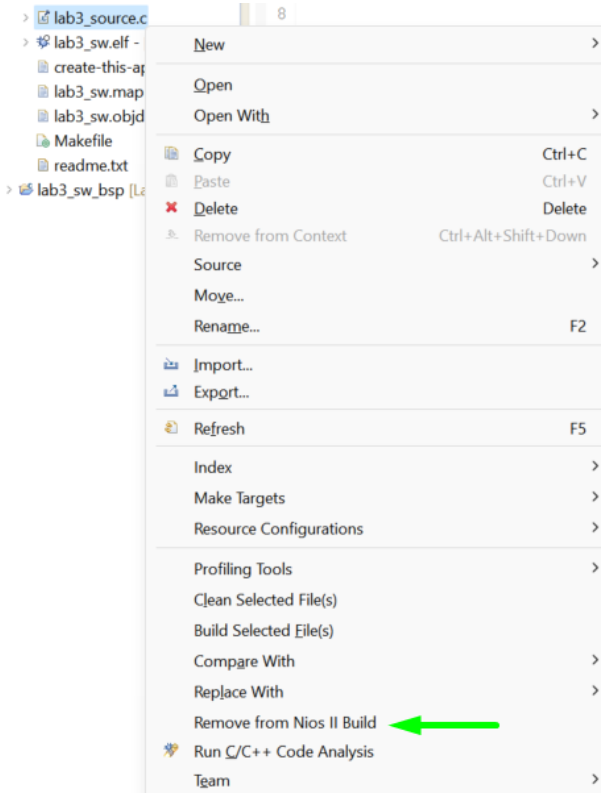
    - Подключите `alt_stdio.h`

      ```
      #include "sys\alt_stdio.h"
      ```

    - Замените все операторы printf() на alt_printf()

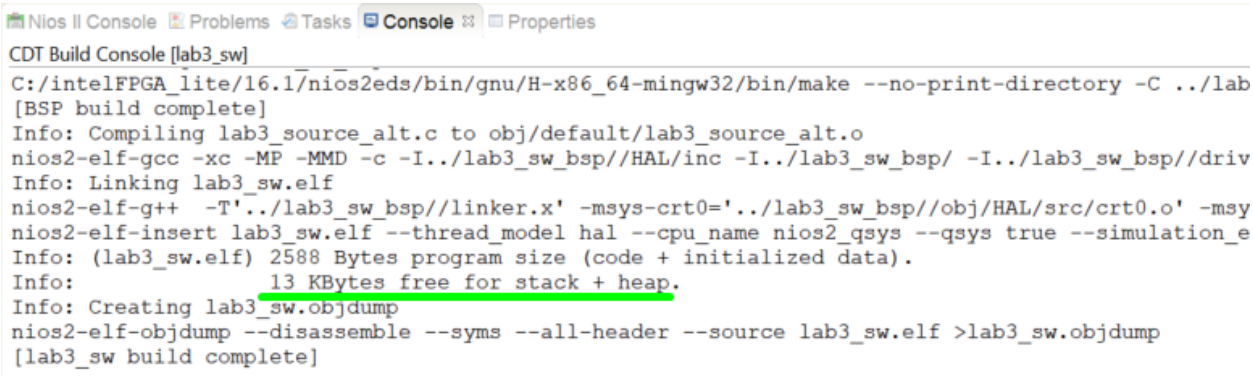
12. Сохраните его под именем lab3_source_alt.c

13. Файл lab3_source.c исключите из компиляции.



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

15. Результаты компиляции показывают, что требования к объему памяти для исходного кода программ и инициализационных данных сократилось до 2588 байт. При этом под стек и данные осталось 13К байт.



16. Выберите проект *lab3_sw_bsp*,

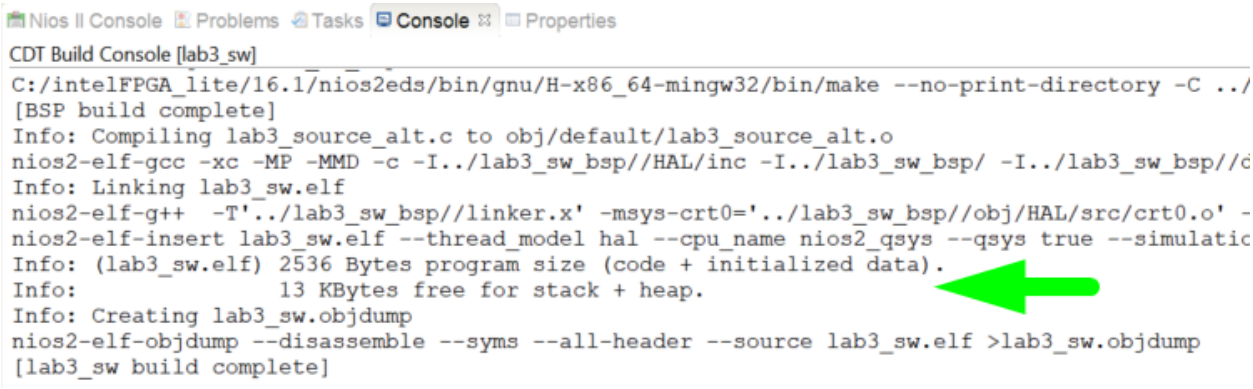
    - нажмите правую клавишу мыши, выберите команду Properties,
    - переключитесь на закладку NIOS II BSP Properties и установите Optimization Level равным size. Нажмите кнопку Apply, затем OK.

17. Выберите проект *lab3_sw*,

    - нажмите правую клавишу мыши, выберите команду Properties,
    - переключитесь на закладку NIOS II Application Properties и установите Optimization Level равным size. Нажмите кнопку кнопку Apply, затем OK.

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

19. Результаты компиляции показывают, что требования к объему памяти для исходного кода программ и инициализационных данных сократилось до 2536 байт. При этом под стек и данные осталось 13Кбайт.



20. Если необходимо: с помощью команды меню Tools=> QuartusII Programmer запрограммируйте плату.

21. Для проекта lab3_sw выполните команду Run => Run as => NiosII Hardware и убедитесь, что проект работает и использованные Вами ранее функции отладки доступны.

Упражнение 3 завершено.