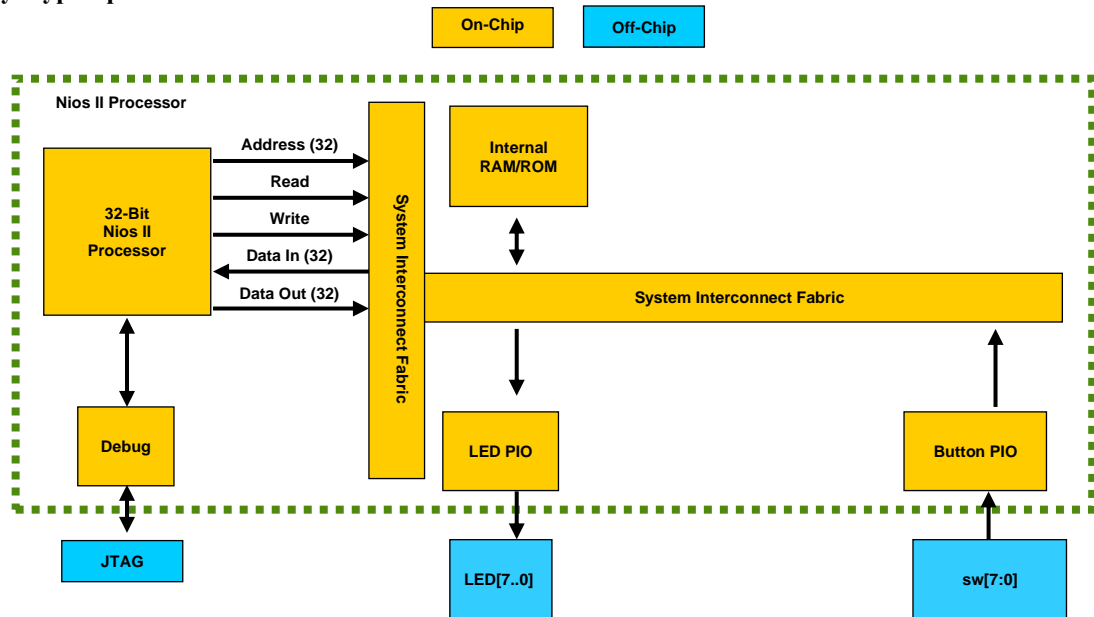


Задание labn_2

Введение:

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

Структура проекта:**Алгоритм работы проекта:**

Под управлением процессора NIOSII обеспечивается:

- Опрос состояния переключателя sw[0] (все остальные переключатели в 0)
- Борьба с дребезгом контактов
- При каждом переключении sw[0] из 1 в 0 - 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 Quartus Prime
2. В меню **File** менеджера пакета, укажите **New Project Wizard...**
3. На экране появится окно введения - **Introduction** (если оно не было отключено). Нажмите кнопку **next**.
4. В появившемся окне введите следующие данные:

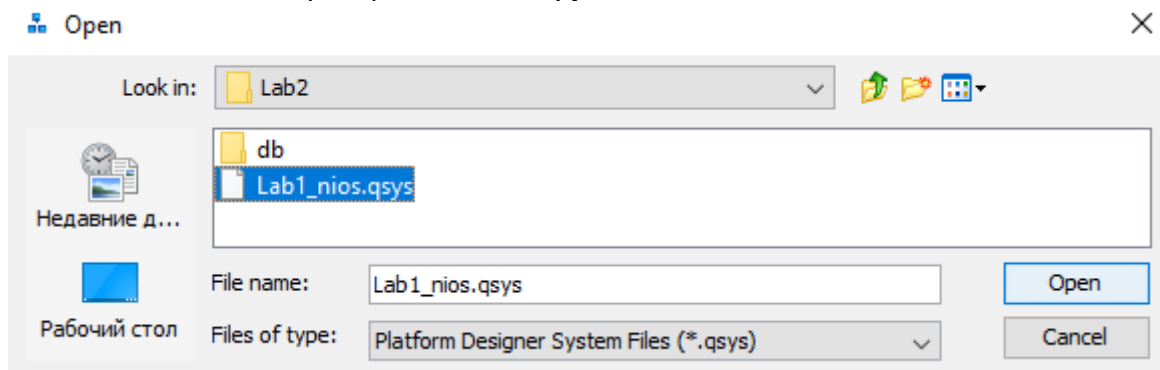
What is the working directory for this project? Рабочая папка (с помощью браузера найдите рабочую папку проекта)	C:\Intel_trn\Q_NIOS\Lab2
What is the name of this project? Имя проекта	Lab2
What is the name of the top-level design entity for this project? Имя модуля верхнего уровня в иерархии проекта.	Lab2

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. Из папки C:\Intel_trn\Q_NIOS\Lab1 скопируйте файл Lab1_nios.qsys в рабочую папку текущего проекта C:\Intel_trn\Q_NIOS\Lab2
2. Выполните команду **Tools => Platform Designer**. Будет запущен **Platform Designer**
3. В появившемся окне выберите файл Lab1_nios.qsys



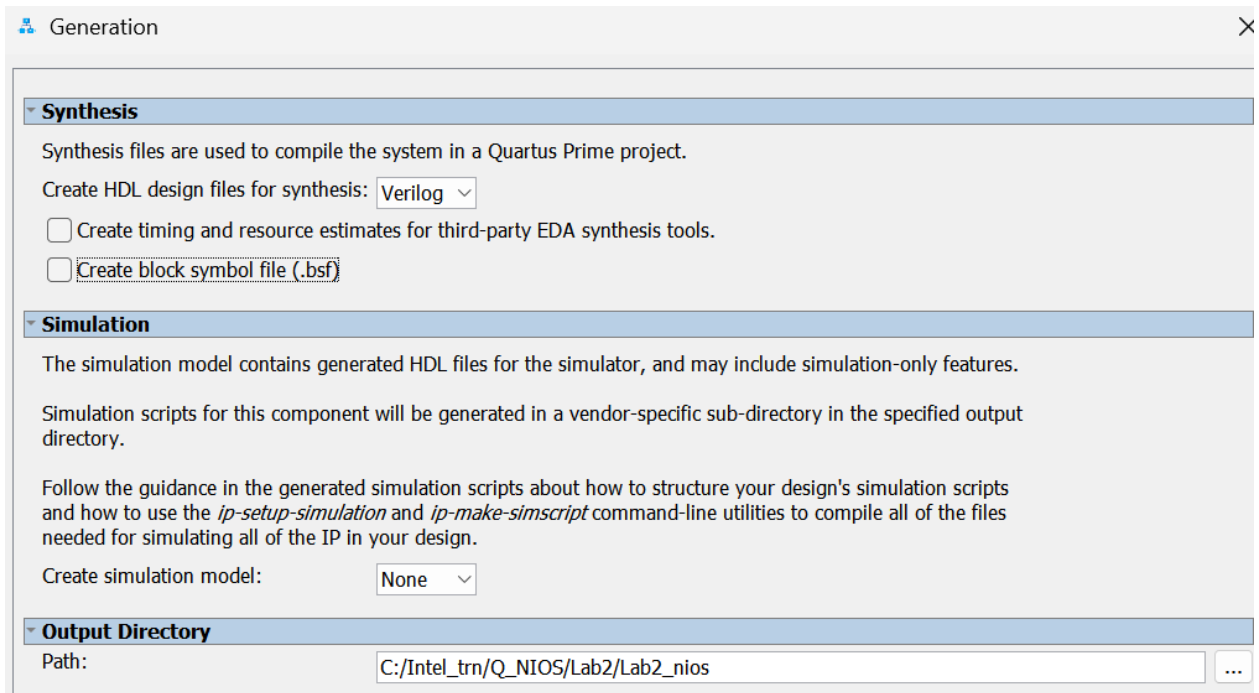
4. Откроется закладка System Contents, в которой будет отображена система, созданная в Lab1
5. Выполните команду **File=>Save as** и сохраните систему под именем **Lab2_nios.qsys**
6. Конфигурация процессора NIOSII
 - ✓ Выделите модуль nios2_PD => нажмите правую клавишу мыши => выберите команду Edit... => откроется окно задания параметров модуля
 - ✓ Переключитесь на закладку JTAG Debug и установите параметр Include JTAG Debug.
 - ✓ Нажмите кнопку Finish.
7. Соедините data_master и instruction_master компонента nios2_PD с входом debug_mem_slave компонента nios2_PD.
8. Выполните автоматическое распределение адресного пространства системы: **System=>Assign base Addresses**
9. Внешний вид созданной системы, закладка System Contents

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk	Clock Source	clk	exported		
		clk_in	Clock Input	reset	[clk_in]		
		clk_in_reset	Reset Input	Double-click to export	clk		
		clk	Clock Output	Double-click to export			
<input checked="" type="checkbox"/>		onchip_memory	On-Chip Memory (RAM or ROM) Intel ...				
		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]		
<input checked="" type="checkbox"/>		nios2_PD	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]		
		debug_reset_request	Reset Output	Double-click to export	[clk]		
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]		
		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]		
						0x8800	0x8fff
<input checked="" type="checkbox"/>		pio_LED	PiO (Parallel I/O) Intel FPGA IP				
		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
<input checked="" type="checkbox"/>		pio_SW	PiO (Parallel I/O) Intel FPGA IP				
		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

10. Закладка Address Map

System Contents	Address Map	Interconnect Requirements
System: Lab2_nios		
	nios2_PD.data_master	nios2_PD.instruction_master
nios2_PD.debug_mem_slave	0x8800 - 0x8fff	0x8800 - 0x8fff
onchip_memory.s1	0x4000 - 0x7fff	0x4000 - 0x7fff
pio_LED.s1	0x9010 - 0x901f	
pio_SW.s1	0x9000 - 0x900f	

11. Выполните команду **File=>Save**
12. Откройте окно настройки процедуры формирования описания системы: **Generate=>Generate HDL**.



- ✓ нажмите кнопку Generate.
- ✓ Создание HDL описания системы должно завершиться без ошибок и предупреждений.
- ✓ Нажмите кнопку Close.

13. В окне Platform Designer нажмите кнопку Finish
14. Появится окно, напоминающее о том, что к проекту необходимо подключить файл Lab2_nios.qip с описанием созданной системы.
15. В пакете QR выполните команду меню Project => Add/Remove Files in Project
16. В появившемся окне найдите (в папке C:\Intel_trn\Q_NIOS\Lab2\Lab2_nios\synthesis) и подключите к проекту файл Lab2_nios.qip

Часть 3 – создание файла верхнего уровня иерархии в описании проекта

1. Создайте в текстовом редакторе файл (имя файла Lab2.sv) верхнего уровня в иерархии проекта (для этого целесообразно использовать файл Lab2_nios_inst.v из папки C:\Intel_trn\Q_NIOS\Lab2\Lab2_nios)
 - a. Пример файла приведен на рисунке

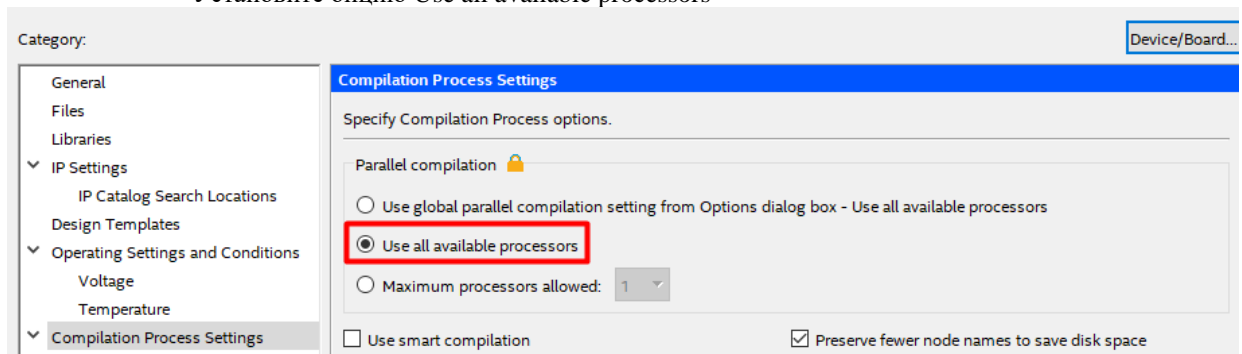
```

1  module Lab2 (
2      input bit clk,          // Clock
3      input bit [7:0]sw,     // data in
4      input bit pbb,         // System reset active low
5      output bit [7:0] led
6  );
7  Lab2_nios u0 (
8      .clk_clk      (clk),    // clk.clk
9      .reset_reset_n (pbb),   // reset.reset_n
10     .led_export    (led),    // led.export
11     .sw_export     (sw)      // pbb.export
12  );
13 endmodule

```

Часть 3 – Полная компиляция проекта

1. Выполните команду Assignments=>Settings => Compilation Process Settings
 - ✓ Установите опцию Use all available processors



2. 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
3. Проверка синтаксиса проекта.
 - ✓ Выполните команду Processing=>Start=>Start Analysis and Elaboration
 - ✓ Компиляция должна завершиться без ошибок.
4. Назначение выводов проекта.
 - a. Запустите редактор назначения выводов (Pin Planner): Assignment=>Pin Planner.
 - b. Назначьте выводы так, как показано на рисунке ниже

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength
in clk	Input	PIN_23	1	B1_N0	3.3-V LVTTTL		8mA (default)
out led[7]	Output	PIN_65	4	B4_N0	2.5 V		8ma
out led[6]	Output	PIN_66	4	B4_N0	2.5 V		8ma
out led[5]	Output	PIN_67	4	B4_N0	2.5 V		8ma
out led[4]	Output	PIN_68	4	B4_N0	2.5 V		8ma
out led[3]	Output	PIN_69	4	B4_N0	2.5 V		8ma
out led[2]	Output	PIN_70	4	B4_N0	2.5 V		8ma
out led[1]	Output	PIN_71	4	B4_N0	2.5 V		8ma
out led[0]	Output	PIN_72	4	B4_N0	2.5 V		8ma
in pbb	Input	PIN_58	4	B4_N0	2.5 V		8mA (default)
in sw[7]	Input	PIN_88	5	B5_N0	3.3-V LVTTTL		8mA (default)
in sw[6]	Input	PIN_89	5	B5_N0	3.3-V LVTTTL		8mA (default)
in sw[5]	Input	PIN_90	6	B6_N0	3.3-V LVTTTL		8mA (default)
in sw[4]	Input	PIN_91	6	B6_N0	3.3-V LVTTTL		8mA (default)
in sw[3]	Input	PIN_49	3	B3_N0	3.3-V LVTTTL		8mA (default)
in sw[2]	Input	PIN_46	3	B3_N0	3.3-V LVTTTL		8mA (default)
in sw[1]	Input	PIN_25	2	B2_N0	3.3-V LVTTTL		8mA (default)
in sw[0]	Input	PIN_24	2	B2_N0	3.3-V LVTTTL		8mA (default)

5. С помощью Timing Analyzer или в текстовом редакторе создайте файл (**Lab2.sdc**) с требованиями к временным параметрам проекта. Пример файла приведен на рисунке

```

#####
# 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 {sw[0]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[1]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[2]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[3]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[4]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[5]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[6]}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {sw[7]}]

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}]

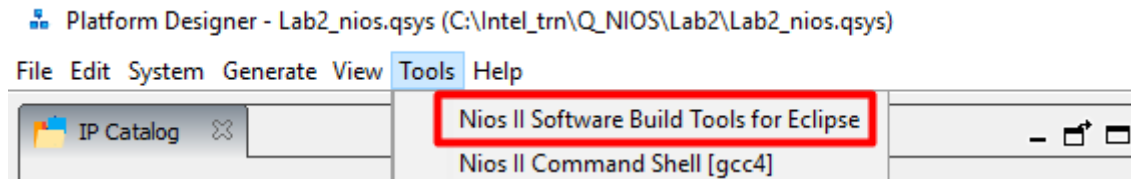
```

6. Подключите файл Lab2.sdc к проекту.
 - ✓ Выполните команду Assignment=>Settings
 - ✓ В разделе Timing Analyzer добавьте файл Lab2.sdc к проекту.
7. В окне менеджера пакета QuartusII, с помощью команды **Processing => Start Compilation** осуществите полную компиляцию проекта.
8. Компиляция должна завершиться без ошибок. Все требования к временным параметрам должны быть выполнены.
9. Обратите внимание на предупреждение о том, что память процессора остается незаполненной при компиляции аппаратной части проекта.

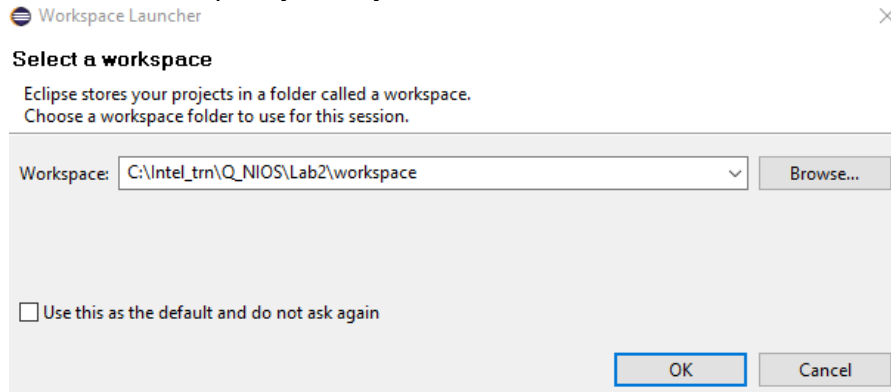
▲ 127003 Can't find Memory Initialization File or Hexadecimal (Intel-Format) File C:/Intel_trn/Q_NIOS/Lab2/onchip_mem.hex -- setting all initial values to 0

Часть 4 – Создание программной части проекта

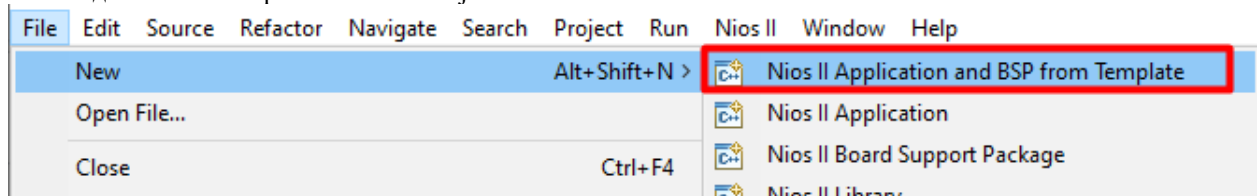
- Из приложения PD запустите оболочку для разработки/отладки программ :
 ✓ Tools=> NiosII Software Build Tools for Eclipse



- В появившемся окне задайте рабочую папку



- Выполните команду File=>New=>NIOS II Application and BSP from Template. Будет запущен помощник создания нового проекта – New Project Wizard

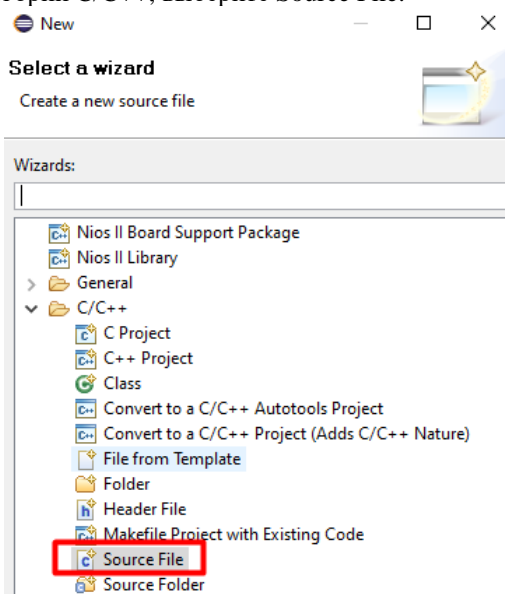


- В появившемся окне помощника:

- ✓ В разделе Select Target Hardware: с помощью браузера найдите в рабочей папке и укажите файл lab2_nios.sopcinfo – файл с описанием созданной системы на кристалле.
- ✓ В разделе Select Project Template: выберите Blank Project
- ✓ В разделе Name: введите название проекта – lab2_sw
- ✓ Нажмите кнопку Finish.

- Выполните команду File=>New=>Other.

- В появившемся окне, в категории C/C++, выберите Source File.



- ✓ Нажмите кнопку Next.

- В окне New source file

- ✓ Поле Source folder: с помощью браузера найдите укажите папку lab2_sw,

- ✓ Поле Source file: введите название файла - lab2_source.c;
- ✓ Поле Template: укажите Default C source template.

Source folder:	lab2_sw
Source file:	lab2_source.c
Template:	Default C source template

21. Нажмите кнопку Finish.
22. Будет создан и открыт в текстовом редакторе новый файл.
23. Введите текст программы на языке Си:

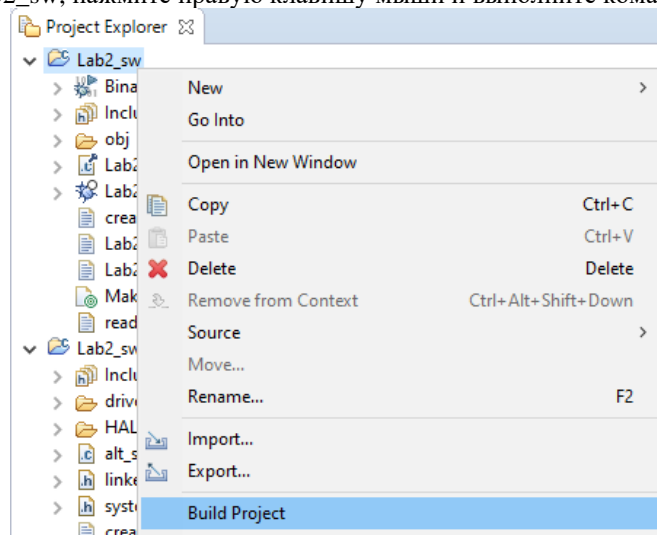
```

1  #include "system.h"
2  #include "altera_avalon_pio_regs.h"
3  #include <unistd.h>
4
5  #define EQ_ONE 0x01      // Value read from sw PIO
6  #define DEBOUNCE 30000  // Time in microseconds to wait for switch debounce
7
8  int main(void) {
9      int sw ;             // Use to hold sw value
10     int led = 0x00;       // Use to write to led
11
12     while (1)
13     {                     // Read buttons via pio
14         sw = IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE);
15
16         if (sw != EQ_ONE) // if sw[0] is 0
17         {
18             if (led >= 0x80 || led==0x00)
19                 led = 0x01; // reset pattern
20             else
21                 led = led << 1; // shift left on board (led0 is far right)
22
23             IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE,~led); // Write new value to pio
24
25             // Switch debounce routine
26             usleep (DEBOUNCE);
27             while (sw != EQ_ONE) // wait for sw =1
28                 sw = IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE);
29             usleep (DEBOUNCE);
30         }
31     }
32 }

```

- ✓ Сохраните его.

24. Выберите папку Lab2_sw, нажмите правую клавишу мыши и выполните команду Build Project.



- ✓ При успешном завершении процесса, в окне Console появится сообщение

```

nios2-elf-g++ -T'../Lab2_sw_bsp//linker.x' -msys-crt0='../Lab2_sw_bsp//obj/HAL/src/crt0.o' -msy
nios2-elf-insert Lab2_sw.elf --thread model hal --cpu name nios2 PD --qsys true --simulation_ena
Info: (Lab2_sw.elf) 4616 Bytes program size (code + initialized data).
Info: 10 KBytes free for stack + heap.
Info: Creating Lab2_sw.objdump
nios2-elf-objdump --disassemble --syms --all-header --source Lab2_sw.elf >Lab2_sw.objdump
[Lab2_sw build complete]

```

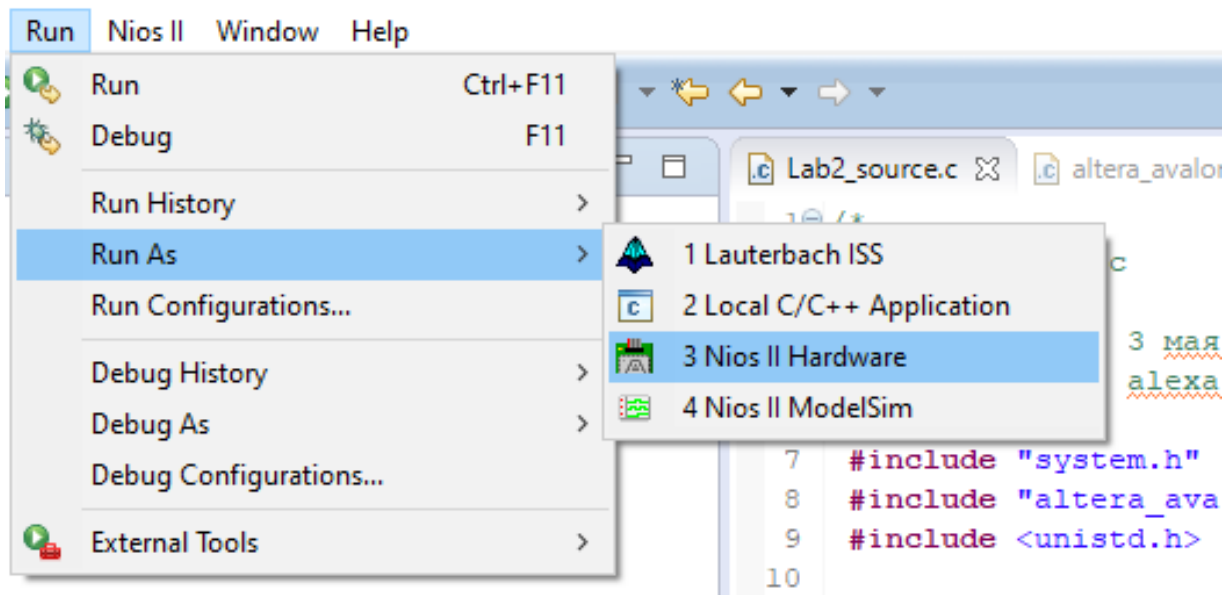
14:35:57 Build Finished (took 7s.956ms)

Часть 5 – Конфигурирование FPGA

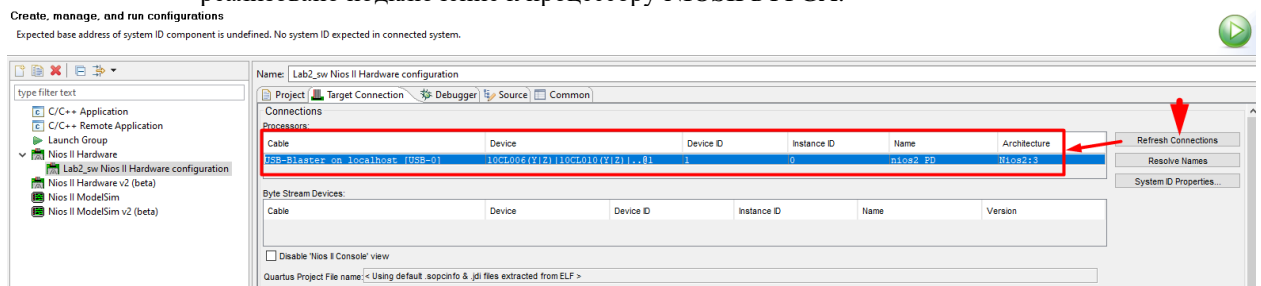
1. Подключите плату miniDilabCIV к ПК.
2. Включите питание платы.
3. Все переключатели sw[7:1] установите в 0, а переключатель SW[0] установите в положение 1
4. В пакете QR выполните команду **Tools=>Programmer**
5. Откроется окно управления конфигурированием СБИС.
 - ✓ Установите средство конфигурирования FPGA
 - ✓ Выберите файл для конфигурирования
 - ✓ Включите опцию **Program/Configure**
 - ✓ Нажмите кнопку **Start**.
6. В окне Progress будет отображаться статус процедуры конфигурирования.

Часть 6 – Загрузка ПО

1. В приложении Eclipse (IDE для разработки ПО) выполните команду **Run=>Run as => NiosII Hardware**



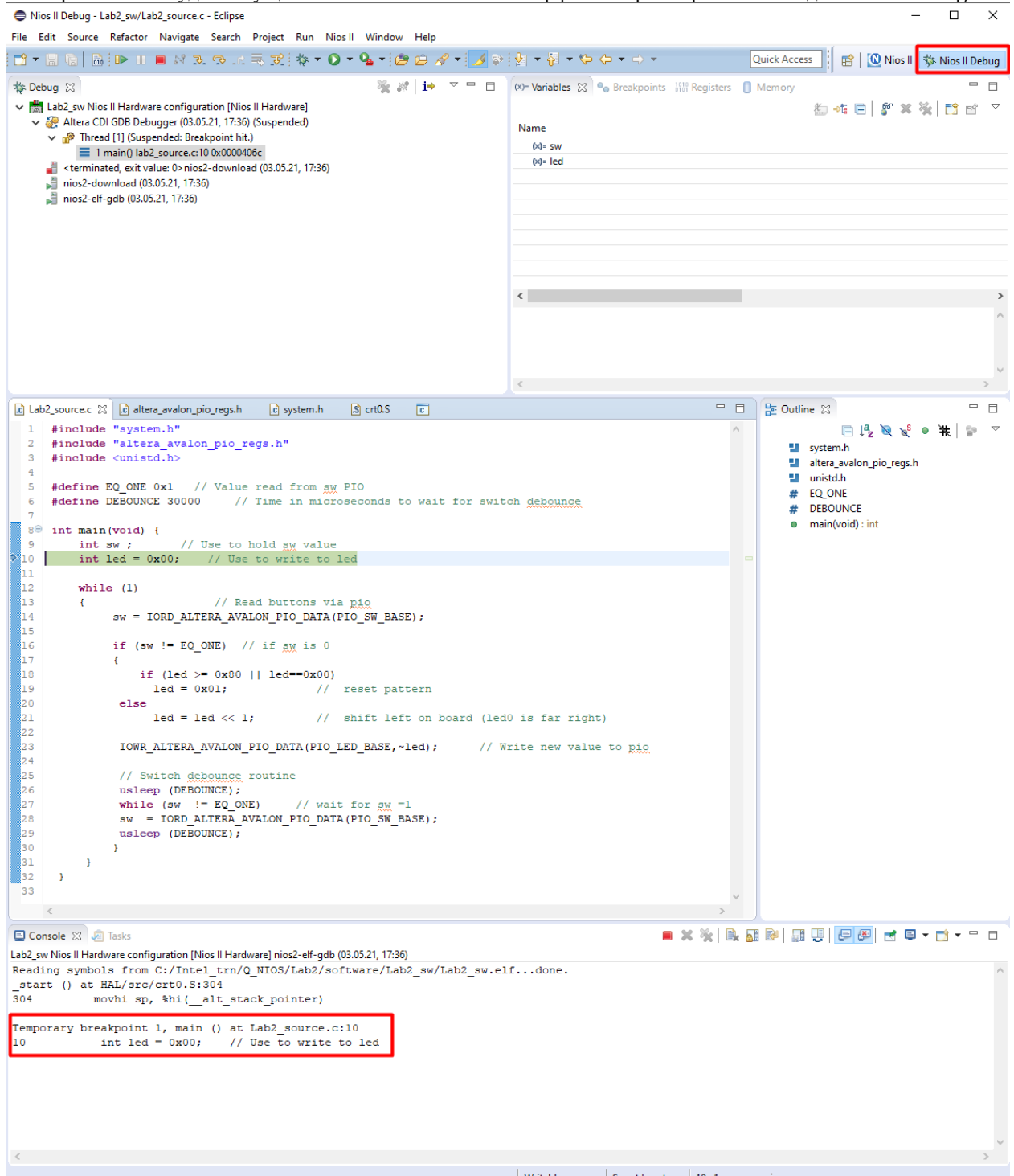
2. В появившемся окне:
 - ✓ Выберите закладку Target Connection
 - ✓ Нажмите кнопку Refresh Connections - будет установлен имеющийся у Вас JTAG кабель и реализовано подключение к процессору NIOSII в FPGA.



3. Нажмите кнопку Apply, а затем Close.
4. В приложении Eclipse (IDE для разработки ПО) еще раз выполните команду **Run=>Run as => NiosII Hardware**
 - ✓ ПО будет загружено в память процессора NIOSII в FPGA.
5. Проверьте работу проекта на плате
 - ✓ Переключите SW[0] из 1 в 0 - включится светодиод LED[0]
 - ✓ Несколько раз переключите SW[0] из 0 в 1, а затем в 0
 1. При этом будет включаться один из светодиодов в следующей последовательности led [0] => led[1] => led[2] => ... => led[7] => led[0] =>
 - ✓ Переключите SW[0] в 1.

Часть 7 – Отладка ПО

1. В приложении Eclipse выполните команду **Run=>Debug as => Nios II Hardware**
2. При успешной загрузке приложения появится окно с сообщением - “переключиться GUI к виду ОТЛАДКА?”. Нажмите кнопку Yes.
3. Приложение будет запущено и пользовательский интерфейс открыт в режиме отладки **NiosII Debug**



4. Выполните команду Run=>Resume
6. Проверьте работу проекта на плате
 - ✓ Переключите SW[0] из 1 в 0 - включится светодиод led[0]
 - ✓ Несколько раз переключите SW[0] из 0 в 1, а затем в 0
 - i. При этом будет включаться один из светодиодов в следующей последовательности led [0] =>led[1]=>led[2]=>...=>led[7]=>led[0]=>....
 - ✓ Переключите SW[0] в 1.
7. Выполните команду Run=>Terminate – выполнение программы остановится
8. Выполните команду Run=>Debug – программа будет запущена в режиме отладки

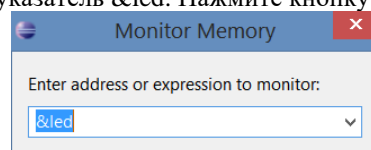
9. Включите (если не включена) опцию отображения номеров строк в исходном тексте.
 - ✓ Меню **Window** => **Preferences**.
 - ✓ В папке **General** => **Editors** => **Text Editor** установите опцию **Show line numbers** и нажмите кнопку **OK**.
10. Установите точки прерывания в строках 26 и 29 (номера строк могут быть другими): поместите курсор на номер строки и дважды щелкните левой клавишей мыши.

```

1  #include "system.h"
2  #include "altera_avalon_pio_regs.h"
3  #include <unistd.h>
4
5  #define EQ_ONE 0x1    // Value read from sw PIO
6  #define DEBOUNCE 30000    // Time in microseconds to wait for switch debounce
7
8  int main(void) {
9      int sw ;          // Use to hold sw value
10     int led = 0x00;    // Use to write to led
11
12     while (1)
13     {
14         // Read buttons via pio
15         sw = IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE);
16
17         if (sw != EQ_ONE) // if sw is 0
18         {
19             if (led >= 0x80 || led==0x00)
20                 led = 0x01;          // reset pattern
21             else
22                 led = led << 1;      // shift left on board (led0 is far right)
23
24             IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, ~led);    // Write new value to pio
25
26             // Switch debounce routine
27             usleep (DEBOUNCE);
28             while (sw != EQ_ONE)    // wait for sw =1
29                 sw = IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE);
30             usleep (DEBOUNCE);
31         }
32     }

```

11. Выберите закладку Memory (правый верхний угол окна пакета в режиме отладки)
12. В поле Monitors нажмите левую клавишу мыши и выполните команду Add Memory Monitor
13. В окне Monitor Memory введите указатель &led. Нажмите кнопку OK.



14. В окне Monitor Memory введите указатель &sw. Нажмите кнопку OK.
15. Нажмите кнопку **resume** – процессор будет запущен и будет ожидать переключения sw[0] в 0
16. Переключите sw[0] в 0:
 - a. на плате включится светодиод led[0]
 - b. процессор остановится в точке прерывания на линии 26
 - c. Обратите внимание на закладку Variables

Name	Value
(x)= sw	0
(x)= led	1


Переменная sw отображает (формат int) значение, полученное с sw[7:0]; переменная led (формат int) отображает число, передаваемое на светодиоды led8...led1

- d. Обратите внимание на закладку Memory

Address	0 - 3	4 - 7	8 - B	C - F
00007FE0	00000000	00000000	01000000	F87F0000
00007FF0	2C420000	F87F0000	EFBEADDE	58400000

По адресу переменной *sw* (адрес 4-7) содержится (формат hex) значение, полученное с *sw*[7:0]; по адресу переменной *led* (адрес 8 –B) содержится (формат hex) число, передаваемое на светодиоды *led*7...*led*0

17. Переключите *sw*[0] в 1

18. Нажмите кнопку **resume**  – процессор будет запущен, но останется в точке прерывания на линии 29

19. Обратите внимание

е. Обратите внимание на закладку Variables

Name	Value
(x)= <i>sw</i>	1
(x)= <i>led</i>	1


Переменная *sw* отображает (формат int) значение, полученное с *sw*[0]; переменная *led* (формат int) отображает число, передаваемое на светодиоды *led*8...*led*1

ф. Обратите внимание на закладку Memory

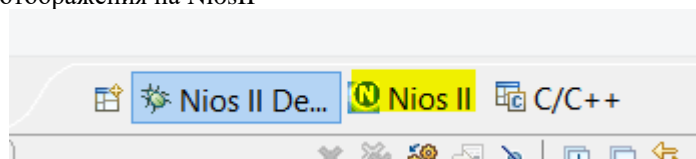
Address	0 - 3	4 - 7	8 - B	C - F
00007FE0	CC400000	01000000	01000000	F87F0000
00007FF0	2C420000	F87F0000	EFBEADDE	58400000

20. По адресу переменной *sw* (адрес 4-7) содержится (формат hex) значение, полученное с *sw*[7:0]; по адресу переменной *led* (адрес 8 –B) содержится (формат hex) число, передаваемое на светодиоды *led*7...*led*0

21. Проведите указанную выше процедуру несколько раз и наблюдайте за переменными.

22. Остановите отладчик – нажмите кнопку 

23. Переключите режим отображения на NiosII



Часть 6 – изменение программы

1. измените текст программы так, чтобы

- использовались указатели (абсолютные адреса элементов PIO_SW и PIO_LED, которые были заданы при создании системы в пакете PD)
- светодиоды включались последовательно начиная с разряда *led*7 (номер включенного светодиода изменялся в сторону младших разрядов при переключении *sw*[0]).

Упражнение 2 завершено.