

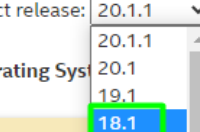
# Задание labn 1

предполагается использование пакета QP Lite версии 16.1...18.1

## Quartus Prime Lite Edition

Release date: November, 2020

Latest Release: v20.1.1



The screenshot shows the 'Operating System' dropdown menu in the Ansys Workbench interface. The '18.1' option is highlighted with a green box. The background shows a list of articles about Ansys Workbench 18.1.

<https://fpgasoftware.intel.com/?edition=lite>

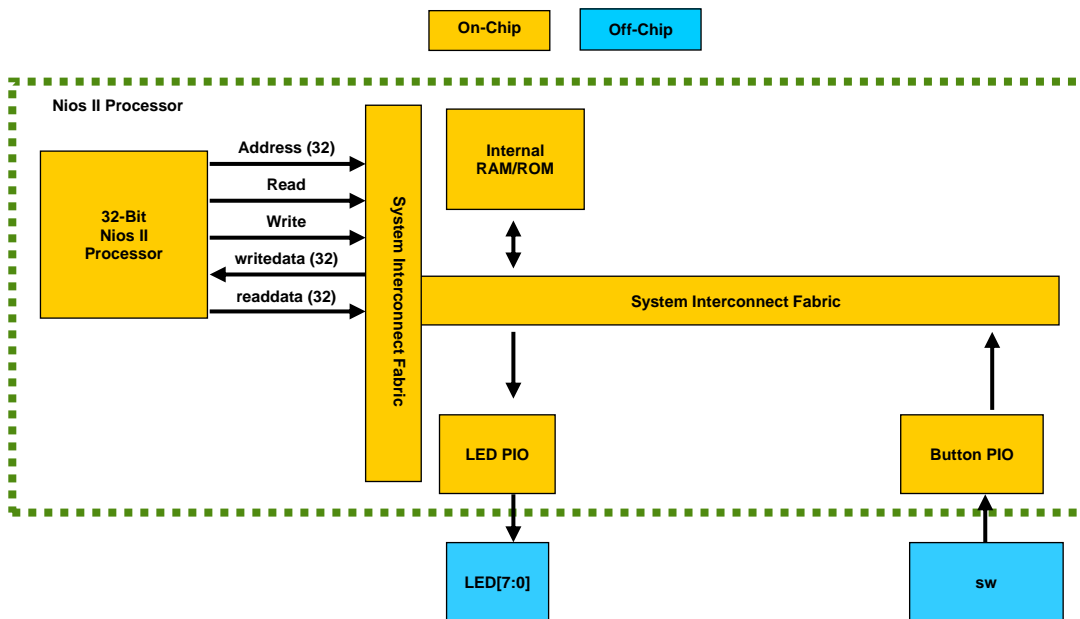
**Введение**

**Цель упражнения** – познакомиться с процедурой реализации «системы на кристалле» – проекта на базе процессора NIOSII, включая следующие этапы:

- ✓ Создание проекта в пакете Quartus Prime (QP)
- ✓ Создание аппаратной части проекта помощью приложения Platform Designer (PD)
- ✓ Создание программной части проекта в рамках оболочки NIOSII IDE
- ✓ Проверка работы проекта на плате

**Структура и алгоритм работы проекта**

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



**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 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\Lab1
What is the name of this project? Имя проекта	Lab1
What is the name of the top-level design entity for this project? Имя модуля верхнего уровня в иерархии проекта.	Lab1

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**.

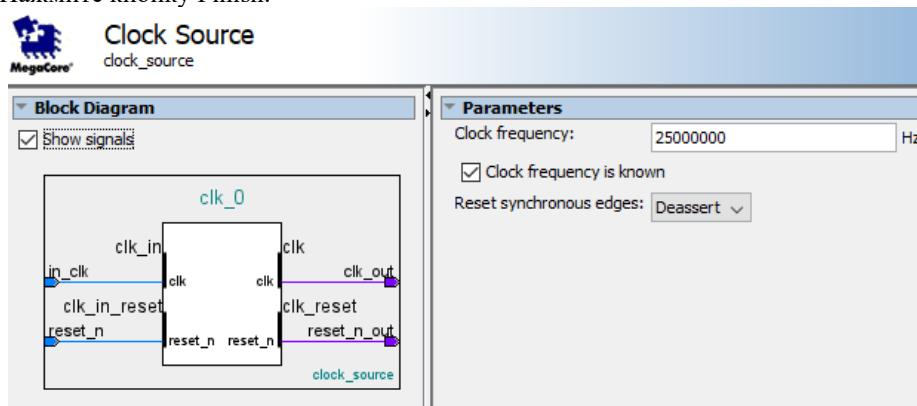
***Проект создан.***

## Часть 2 – Создание аппаратной части проекта

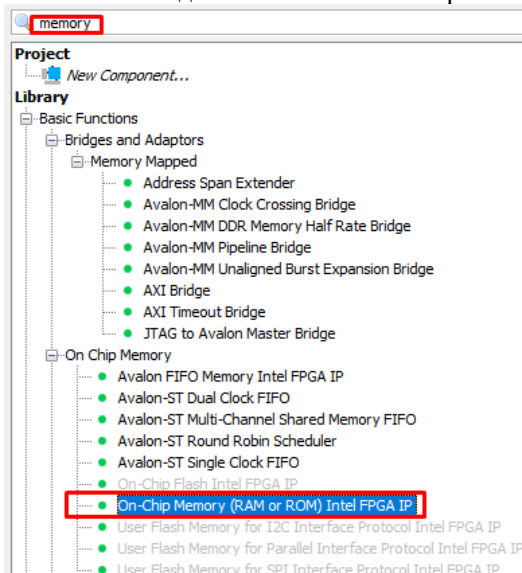
1. Выполните команду **Tools => Platform Designer**. Будет запущен PD и откроется закладка System Contents, в которую по умолчанию будет добавлен компонент source clock

System Contents					
Address Map Interconnect Requirements					
System: unsaved					
Use	Conn...	Name	Description	Export	Clock
<input checked="" type="checkbox"/>		<b>clk_0</b>	Clock Source		
		clk_in	Clock Input	<b>clk</b>	<b>exported</b>  Double-click to export Double-click to export
		clk_in_reset	Reset Input	<b>reset</b>	
		clk	Clock Output		
		clk_reset	Reset Output		

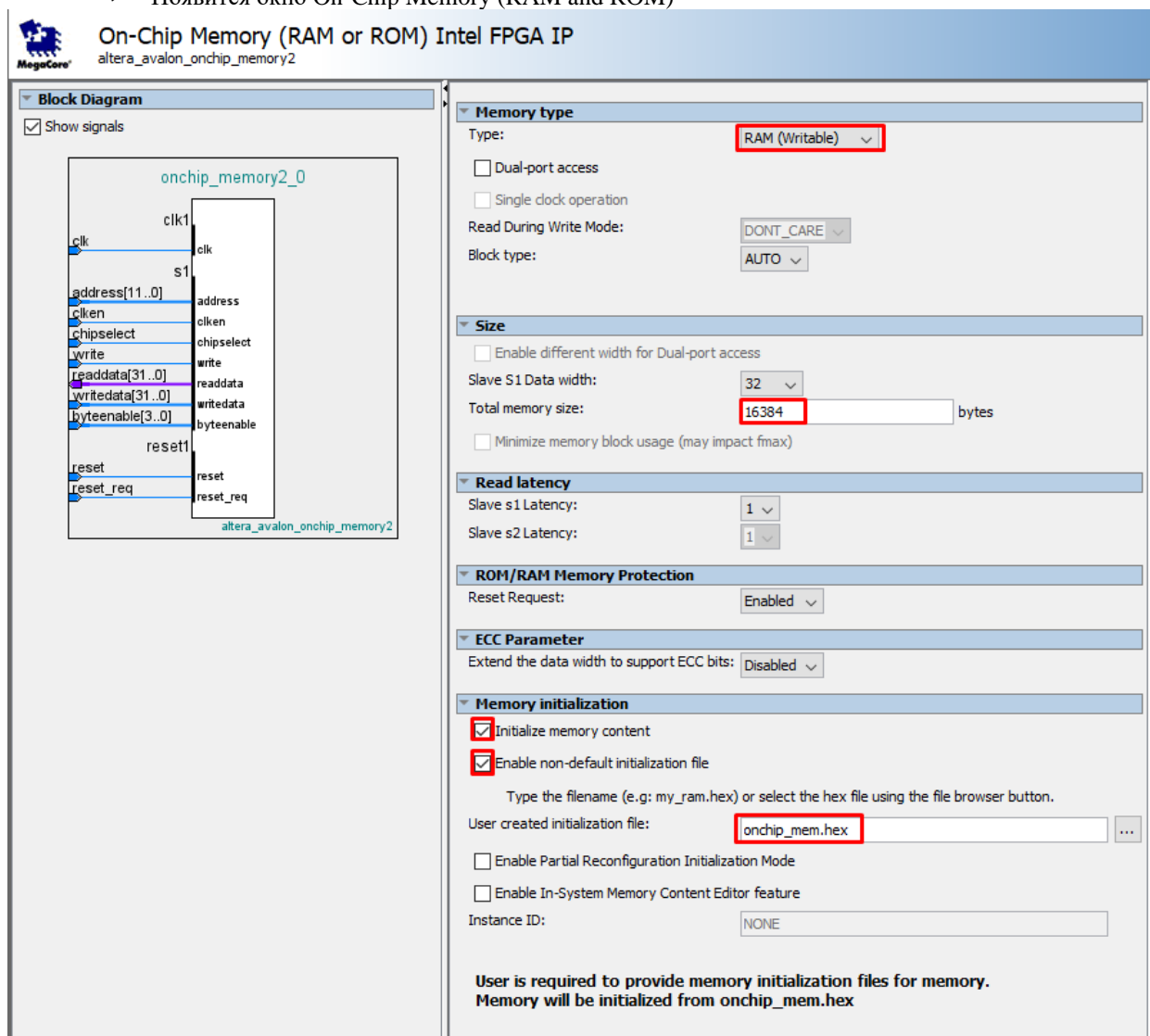
2. Выполните команду **File=>Save as** и сохраните систему под именем **Lab1\_nios**
3. Выделите модуль Clock source => нажмите правую клавишу мыши => выберите команду Edit... => откроется окно задания параметров модуля.
4. В появившемся окне задайте:
  - ✓ частоту тактового сигнала = 25 МГц, что соответствует частоте кварцевого генератора на плате miniDiLaB-CIV.
  - ✓ Параметр Reset synchronous edges: Deassert
  - ✓ Нажмите кнопку Finish.



- ✓ Переименуйте компонент:
    1. в закладке System Contents выберите имя компонента,
    2. нажмите правую клавишу мыши и выберите команду Rename.
    3. Задайте новое имя - clk.
5. Создание, на основе встроенных в FPGA блоков памяти, модуля памяти для команд и данных процессора
    - ✓ В списке IP Catalog приложения PD найдите компонент On-Chip Memory (RAM and ROM).



- ✓ Дважды щелкните компонент левой клавишей мыши
- ✓ Появится окно On-Chip Memory (RAM and ROM)

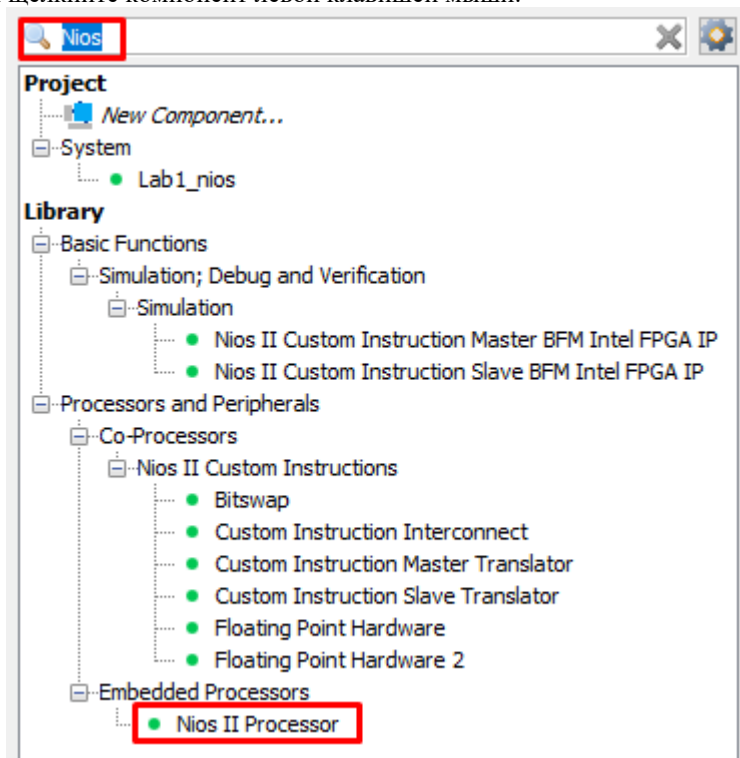


- ✓ В разделе memory type задайте тип памяти – RAM
- ✓ В разделе Size задайте размер памяти 16384 байт.
- ✓ В разделе memory Initialization установите:
  - Режимы так, как показано на рисунке (имя файла д.б. onchip\_mem.hex)
- ✓ Нажмите кнопку Finish. Память для команд и данных процессора создана.  
 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:
  1. на закладке System Contents выберите имя созданного модуля памяти,
  2. нажмите правую клавишу мыши
  3. выберите команду Rename.
  4. Задайте новое имя - onchip\_mem.
- ✓ Соедините выход clk компонента clk с входом clk1 компонента onchip\_mem, а выход clk\_reset компонента clk source с входом reset1 компонента onchip\_mem.

Use	Conn...	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		<b>clk</b>	Clock Source			
		clk_in	Clock Input	<b>clk</b>	<b>exported</b>	
		clk_in_reset	Reset Input	<b>reset</b>	[clk_in]	
		clk	Clock Output	Double-click to export	clk	
		clk_reset	Reset Output	Double-click to export	clk	
<input checked="" type="checkbox"/>		<b>onchip_mem</b>	On-Chip Memory (RAM or ROM) Intel ...			
		clk1	Clock Input	Double-click to export	<b>clk</b>	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	
		reset1	Reset Input	Double-click to export	[clk1]	

## 6. Конфигурация и подключение к системе ядра процессора NIOSII

- ✓ В списке доступных компонентов выберите компонент NIOS II Processor
- ✓ Дважды щелкните компонент левой клавишей мыши.



- ✓ Появится окно конфигурации процессора.
  1. На закладке Main установите тип процессора - NIOSII/e – (простейший вариант процессорного ядра)
  2. На закладке JTAG Debug **отключите** режим Include JTAG Debug (в этой работе отладчик не будет использоваться).
  3. Нажмите кнопку Finish. Ядро процессорного модуля создано и включено в систему.
- Появляющиеся ошибки и предупреждения пока можно проигнорировать.*
- ✓ Переименуйте созданный процессорный модуль:
  1. на закладке System Contents выберите имя созданного модуля,
  2. нажмите правую клавишу мыши
  3. выберите команду Rename.
  4. Задайте новое имя - nios2\_PD.
- ✓ Соедините вход clk компонента nios2\_PD с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента nios2\_PD.
- ✓ Соедините вход s1 компонента onchip\_mem с выходами data\_master и instruction\_master компонента nios2\_PD.

Use	Connections	Name	Description	Export	Clock
<input checked="" type="checkbox"/>		<div> <div>clk</div> <div>Clock Source</div> </div> <div> <div>clk_in</div> <div>Clock Input</div> </div> <div> <div>clk_in_reset</div> <div>Reset Input</div> </div> <div> <div>clk</div> <div>Clock Output</div> </div> <div> <div>clk_reset</div> <div>Reset Output</div> </div>		<div>clk</div> <div>reset</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div>exported</div> <div>[clk_in]</div> <div>clk</div> <div>clk</div>
<input checked="" type="checkbox"/>		<div> <div>onchip_memory</div> <div>On-Chip Memory (RAM or ROM) Intel ...</div> </div> <div> <div>clk1</div> <div>Clock Input</div> </div> <div> <div>s1</div> <div>Avalon Memory Mapped Slave</div> </div> <div> <div>reset1</div> <div>Reset Input</div> </div>		<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div>clk</div> <div>[clk1]</div> <div>[clk1]</div>
<input checked="" type="checkbox"/>		<div> <div>nios2_PD</div> <div>Nios II Processor</div> </div> <div> <div>clk</div> <div>Clock Input</div> </div> <div> <div>reset</div> <div>Reset Input</div> </div> <div> <div>data_master</div> <div>Avalon Memory Mapped Master</div> </div> <div> <div>instruction_master</div> <div>Avalon Memory Mapped Master</div> </div>		<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div>clk</div> <div>[clk]</div> <div>[clk]</div> <div>[clk]</div>
		<div> <div>irq</div> <div>Interrupt Receiver</div> </div> <div> <div>custom_instruction_m...</div> <div>Custom Instruction Master</div> </div>		<div>Double-click to export</div> <div>Double-click to export</div>	<div>[clk]</div>

- ✓ Выделите модуль nios2\_PD => нажмите правую клавишу мыши => выберите команду Edit... => откроется окно задания параметров модуля

✓ На закладке Vectors укажите

1. память для вектора сброса:

2. память для вектора exception:

3. память для вектора Break:

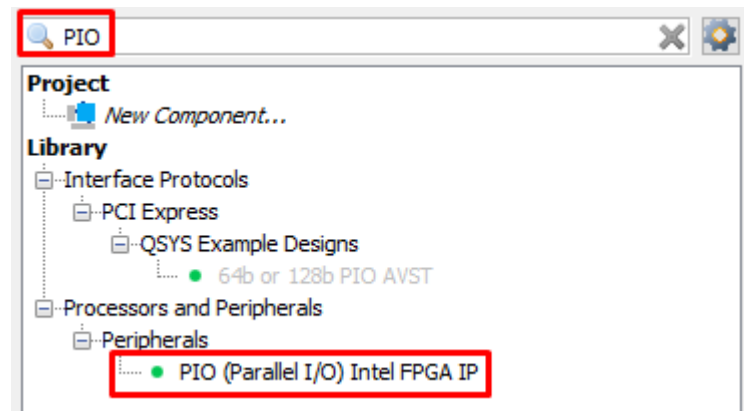
✓ Нажмите кнопку Finish.

В окне сообщений должно остаться одно предупреждение о том, что к процессору не подключен отладчик. Это предупреждение можно проигнорировать.

Type	Path	Message
Warning	1 Warning	
Warning	Lab1_nios.nios2_PD	No Debugger. You will not be able to download or debug programs

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

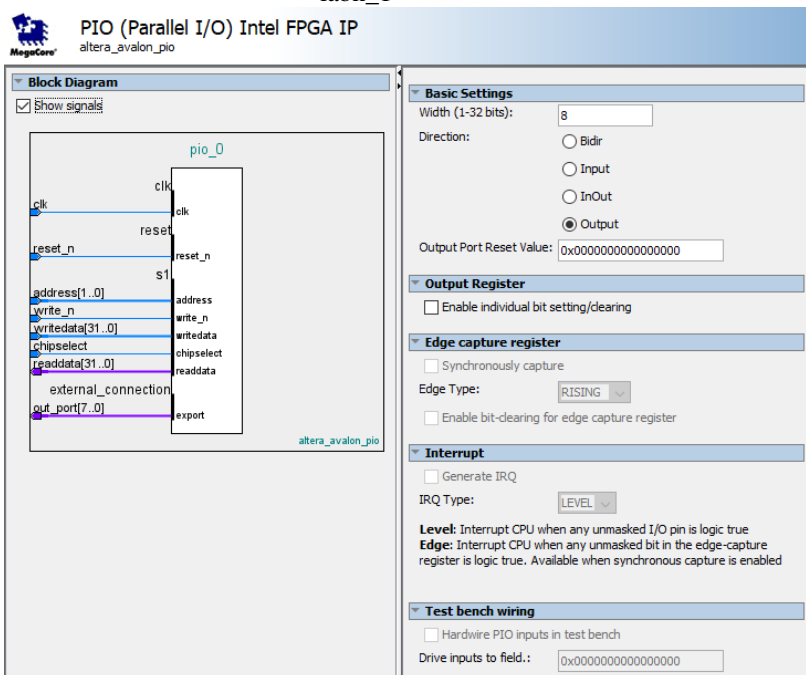
✓ В списке доступных компонентов выберите PIO (Parallel I/O) и дважды щелкните левой клавишей мыши.



✓ 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: на закладке System Contents выберите имя созданного компонента, нажмите правую клавишу мыши и выберите команду Rename. Новое имя - pio\_LED.
- ✓ Соедините вход clk компонента pio\_LED с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента pio\_LED.
- ✓ Соедините вход s1 компонента pio\_LED с выходом data\_master компонента nios2\_PD.
- ✓ Дважды щелкните в строке external\_connection столбца Export, введите имя внешнего вывода создаваемой системы - led

Use	Connections	Name	Description	Export	Clock
<input checked="" type="checkbox"/>		<div>clk</div> <div>clk_in</div> <div>clk_in_reset</div> <div>clk</div> <div>clk_reset</div>	<div>Clock Source</div> <div>Clock Input</div> <div>Reset Input</div> <div>Clock Output</div> <div>Reset Output</div>	<div>clk</div> <div>reset</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div><b>exported</b></div> <div>clk_in</div> <div>clk</div>
<input checked="" type="checkbox"/>		<div>onchip_memory</div> <div>clk1</div> <div>s1</div> <div>reset1</div>	<div>On-Chip Memory (RAM or ROM) Intel ...</div> <div>Clock Input</div> <div>Avalon Memory Mapped Slave</div> <div>Reset Input</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div>clk</div> <div>clk1</div> <div>clk1</div>
<input checked="" type="checkbox"/>		<div>nios2_PD</div> <div>clk</div> <div>reset</div> <div>data_master</div> <div>instruction_master</div> <div>irq</div> <div>custom_instruction_m...</div>	<div>Nios II Processor</div> <div>Clock Input</div> <div>Reset Input</div> <div>Avalon Memory Mapped Master</div> <div>Avalon Memory Mapped Master</div> <div>Interrupt Receiver</div> <div>Custom Instruction Master</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div>clk</div> <div>clk</div> <div>clk</div> <div>clk</div>
<input checked="" type="checkbox"/>		<div>pio_LED</div> <div>clk</div> <div>reset</div> <div>s1</div>	<div>PIO (Parallel I/O) Intel FPGA IP</div> <div>Clock Input</div> <div>Reset Input</div> <div>Avalon Memory Mapped Slave</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	<div>clk</div> <div>clk</div> <div>clk</div>
		external_connection	Conduit	Double-click to export	

Появляющиеся ошибки и предупреждения пока можно проигнорировать.

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

- ✓ В списке доступных компонентов выберите 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.

**PIO (Parallel I/O) Intel FPGA IP**  
altera\_avalon\_pio

**Block Diagram**

☒ Show signals

**Basic Settings**

Width (1-32 bits):

Direction: ☐ Bidir ☒ Input ☐ InOut ☐ Output

Output Port Reset Value:

**Output Register**

☐ Enable individual bit setting/clearing

**Edge capture register**

☐ Synchronously capture

Edge Type:

☐ Enable bit-clearing for edge capture register

**Interrupt**

☐ Generate IRQ

IRQ Type:

**Level:** Interrupt CPU when any unmasked I/O pin is logic true  
**Edge:** Interrupt CPU when any unmasked bit in the edge-capture register is logic true. Available when synchronous capture is enabled

**Test bench wiring**

☐ Hardwire PIO inputs in test bench

Drive inputs to field.:

**Info:** **pio\_0:** PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

- ✓ 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 - **pio\_SW**
- ✓ Соедините вход clk компонента **pio\_SW** с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента **pio\_SW**.
- ✓ Соедините вход s1 компонента **pio\_SW** с выходом data\_master компонента **nios2\_PD**.
- ✓ Появляющиеся ошибки и предупреждения пока можно проигнорировать.
- ✓ Дважды щелкните в строке external\_connection столбца Export, введите имя внешнего вывода создаваемой системы - **sw**
- ✓ Модуль PIO настроен и подсоединен к системе.

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

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		<b>clk</b>	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		
<input checked="" type="checkbox"/>		<b>onchip_memory</b>	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"/>		<b>nios2_PD</b>	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
		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]		
<input checked="" type="checkbox"/>		<b>pio_LED</b>	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]	0x8010	0x801f
		external_connection	Conduit	led			
<input checked="" type="checkbox"/>		<b>pio_SW</b>	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]	0x8000	0x800f
		external_connection	Conduit	sw			

✓ закладка Address Map

System: Lab1_nios Path: clk		
	nios2_PD.data_master	nios2_PD.instruction_master
onchip_memory.s1	0x4000 - 0x7fff	0x4000 - 0x7fff
pio_SW.s1	0x8000 - 0x800f	
pio_LED.s1	0x8010 - 0x801f	

✓ закладка Project Settings. Для отображения установок для созданной системы:

1. Выполните команду View => Hierarchy
2. В окне Hierarchy выберите верхний уровень описания – Lab1\_nios
3. Выполните команду View => Parameters

Parameters

System: Lab1\_nios

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

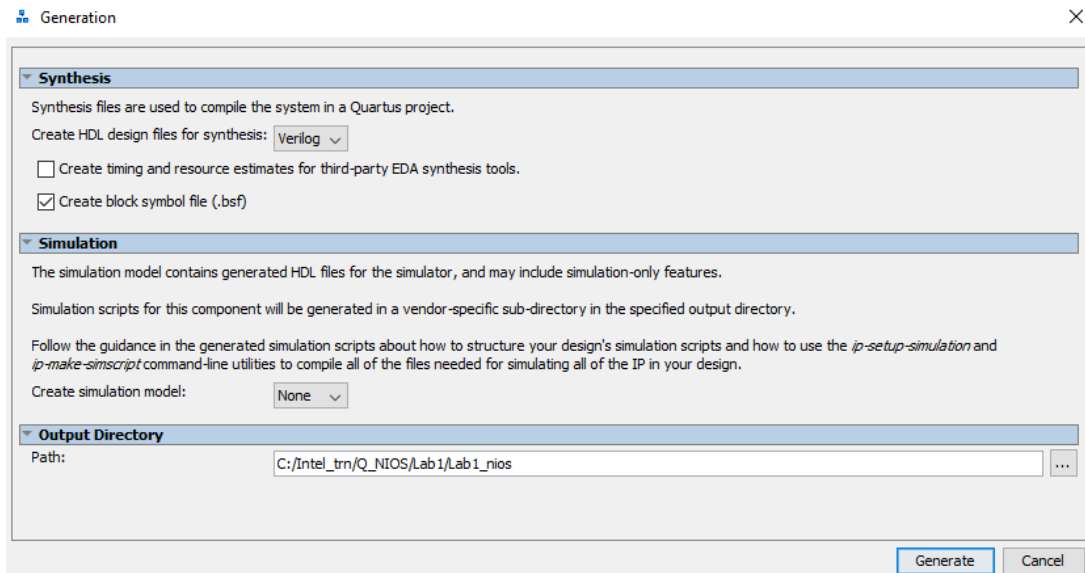
**IP Catalog Settings**

☐ Hide from IP Catalog

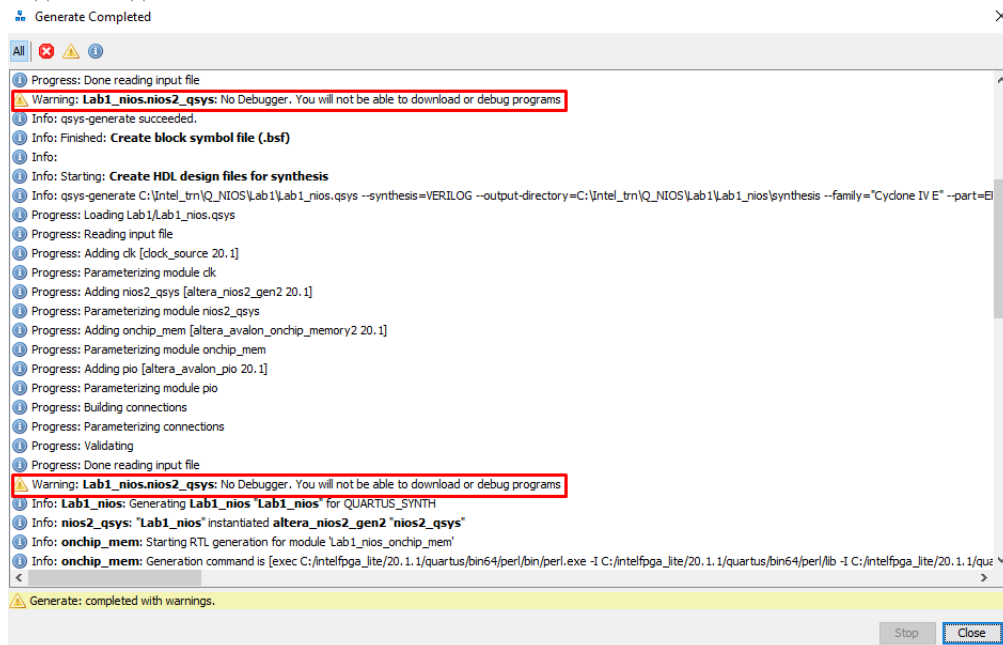
11. Закладка Messages должна содержать одно предупреждение, информирующее о том, что не подключен JTAG Debug модуль (но это было сделано сознательно для данного проекта)

Type	Path	Message
	1 Warning	
	Lab1_nios.nios2_PD	No Debugger. You will not be able to download or debug programs
	1 Info Message	
	Lab1_nios.pio_KEY	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

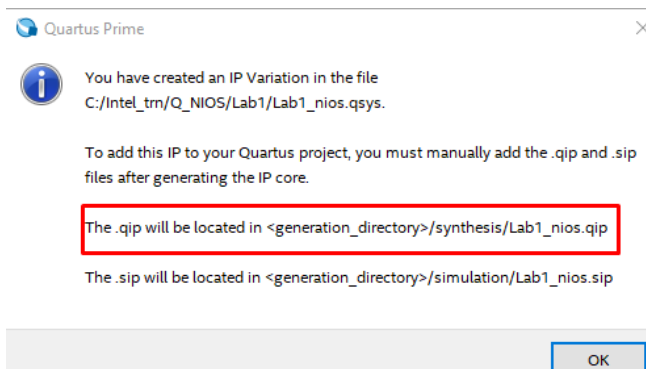
12. Выполните команду File=>Save и откройте окно настройки формирования описания системы:  
Generate=>Generate HDL.



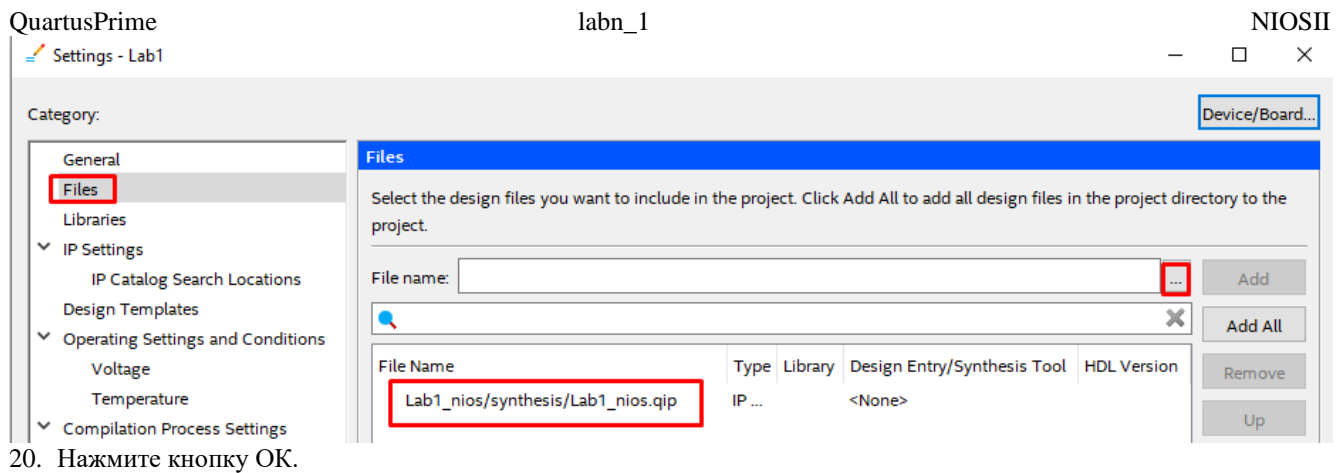
13. Оставьте все значения по умолчанию и нажмите кнопку Generate.
14. Создание HDL описания системы завершится с предупреждениями, касающимися отсутствия JTAG соединения для отладки.



15. Нажмите кнопку Close.
16. В окне Platform Designer нажмите кнопку Finish
17. Появится окно, напоминающее о том, что к проекту необходимо подключить файл Lab1\_nios.qip с описанием созданной системы.



18. В пакете QP выполните команду меню Project => Add\Remove Files in Project
19. В появившемся окне найдите (в папке C:/Intel\_trn/Q\_NIOS/Lab1/Lab1\_nios/synthesis) и подключите к проекту файл Lab1\_nios.qip



### Часть 3 – Интеграция аппаратной части проекта

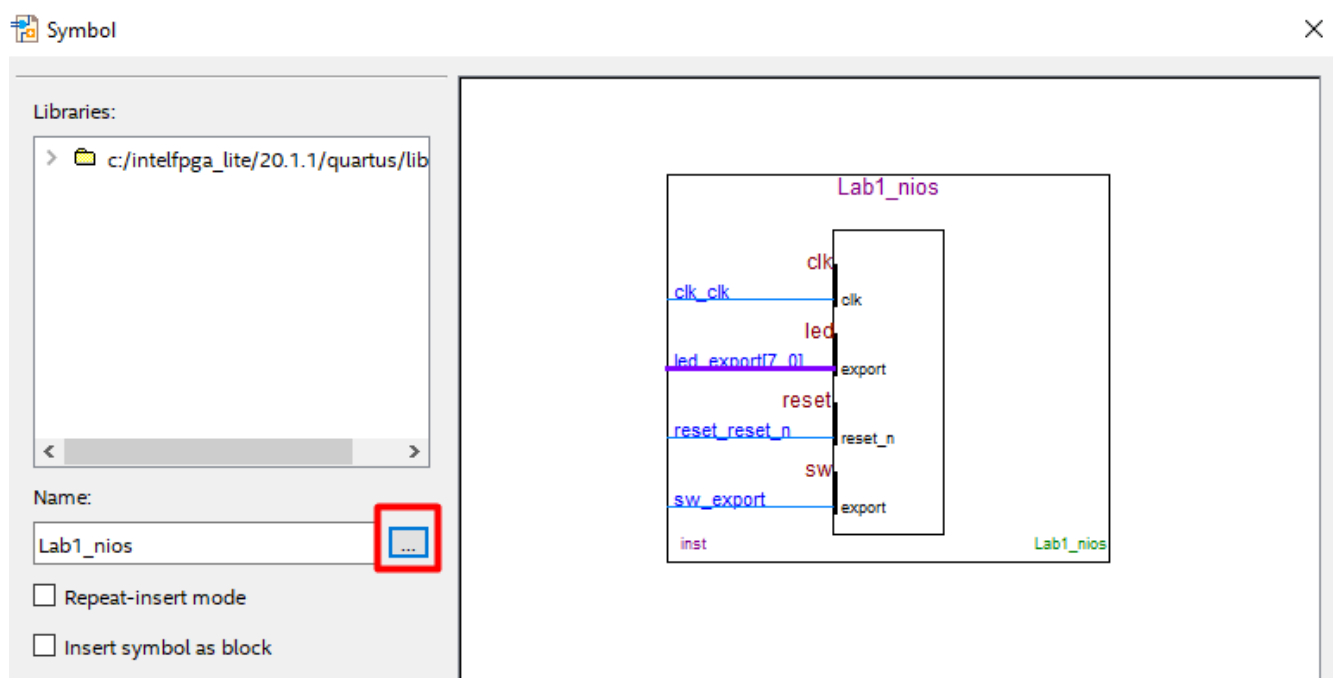
1. Создайте в текстовом редакторе файл (имя файла Lab1.sv) верхнего уровня в иерархии проекта (для этого целесообразно использовать файл Lab1\_nios\_inst.v из папки C:\Intel\_trn\Q\_NIOS\Lab1\Lab1\_nios)

```

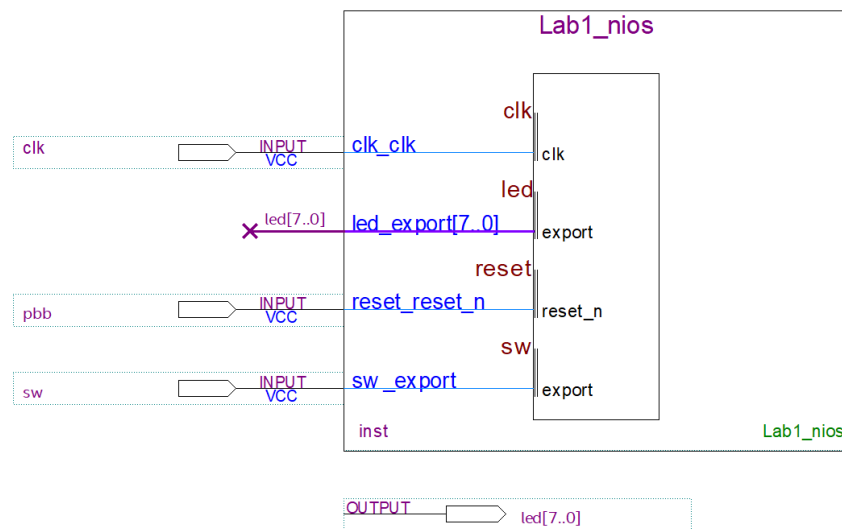
1  module Lab1 (
2      input bit clk, // Clock
3      input bit sw, // data in
4      input bit pbb, // Asynchronous reset active low
5      output bit [7:0] led
6  );
7
8      Lab1_nios u0 (
9          .clk_clk      (clk), // clk.clk
10         .reset_reset_n (pbb), // reset.reset_n
11         .led_export    (led), // led.export
12         .sw_export     (sw)  // pbb.export
13     );
14
15 endmodule

```

2. *Файл верхнего уровня может быть создан и в графическом редакторе*
  - ✓ 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.
  - ✓ Найдите (папка C:\Intel\_trn\Q\_NIOS\Lab1\Lab1\_nios) и укажите символ созданной системы.

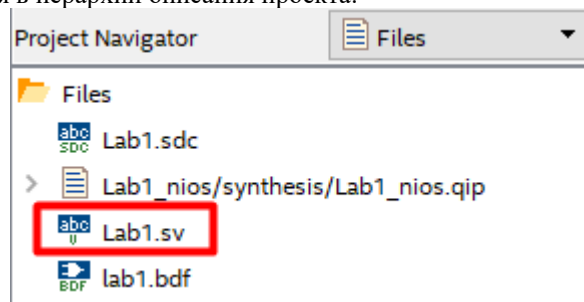


- ✓ Выберите этот символ, нажмите кнопку OK, затем разместите его в поле графического редактора, нажав левую клавишу мыши.
- ✓ Введите схему представленную на рисунке.



- ✓ Сохраните схему под именем lab1.bdf

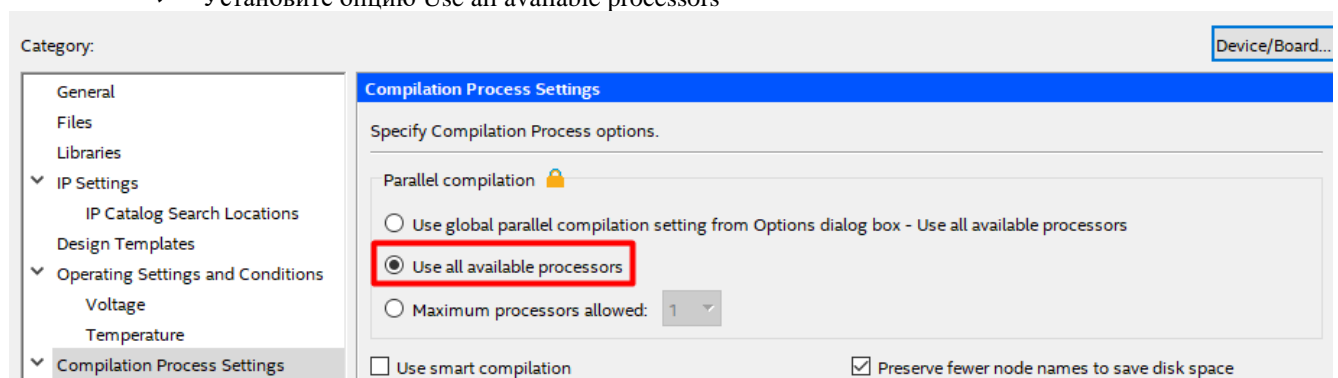
3. Выбор файла верхнего уровня в иерархии описания проекта:



- ✓ При использовании тестового описания – установите файл Lab1.sv как файла верхнего уровня.
  - i. При этом файл Lab1.bdf следует удалить из проекта (Remove File from project)
- ✓ При использовании схемного ввода – установите файл Lab1.bdf как файла верхнего уровня.
  - i. При этом файл Lab1.sv следует удалить из проекта (Remove File from project)

4. Выполните команду Assignments=>Settings => Compilation Process Settings

- ✓ Установите опцию Use all available processors



5. Проверка синтаксиса проекта.

- ✓ Выполните команду Processing=>Start=>Start Analysis and Elaboration
- ✓ Компиляция должна завершиться без ошибок.

6. Проверка структуры проекта

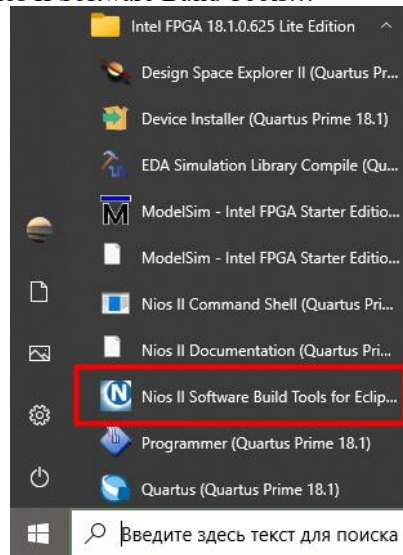
- ✓ Выполните команду Tools => Netlist viewer => RTL Viewer
- ✓ Убедитесь в том, что полученная структура соответствует структуре, приведенной на рисунке.



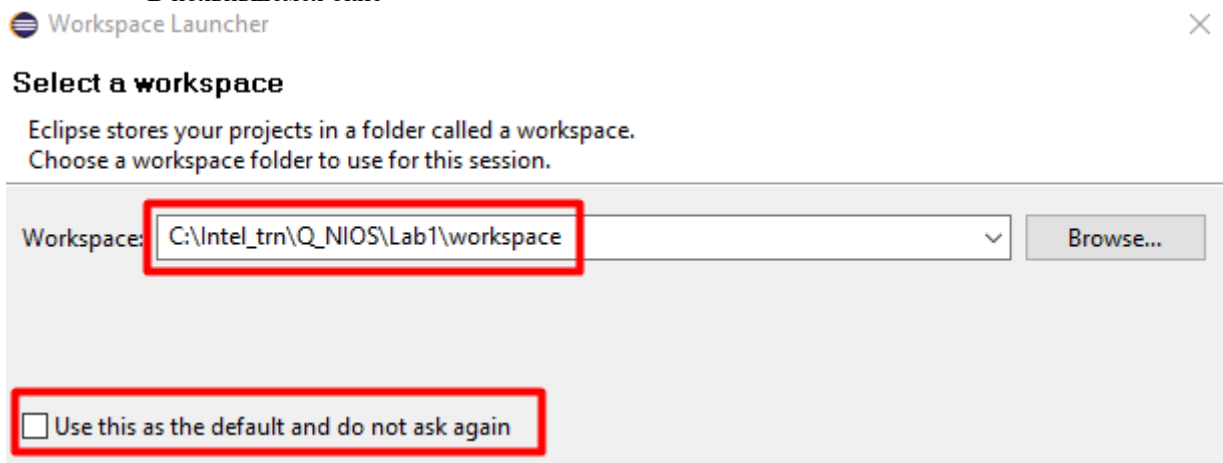
### Часть 3 – Создание программной части проекта

1. Запустите оболочку для разработки/отладки программ - NIOSII IDE:

✓ Start (пуск)=> ... => Nios II Software Build Tools...

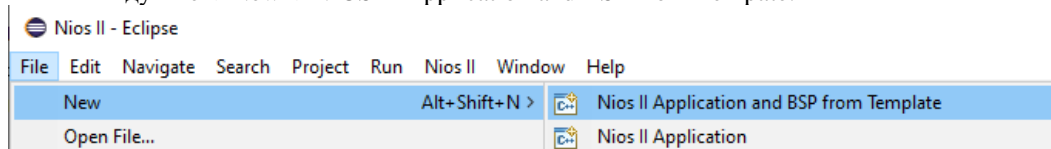


✓ В появившемся окне

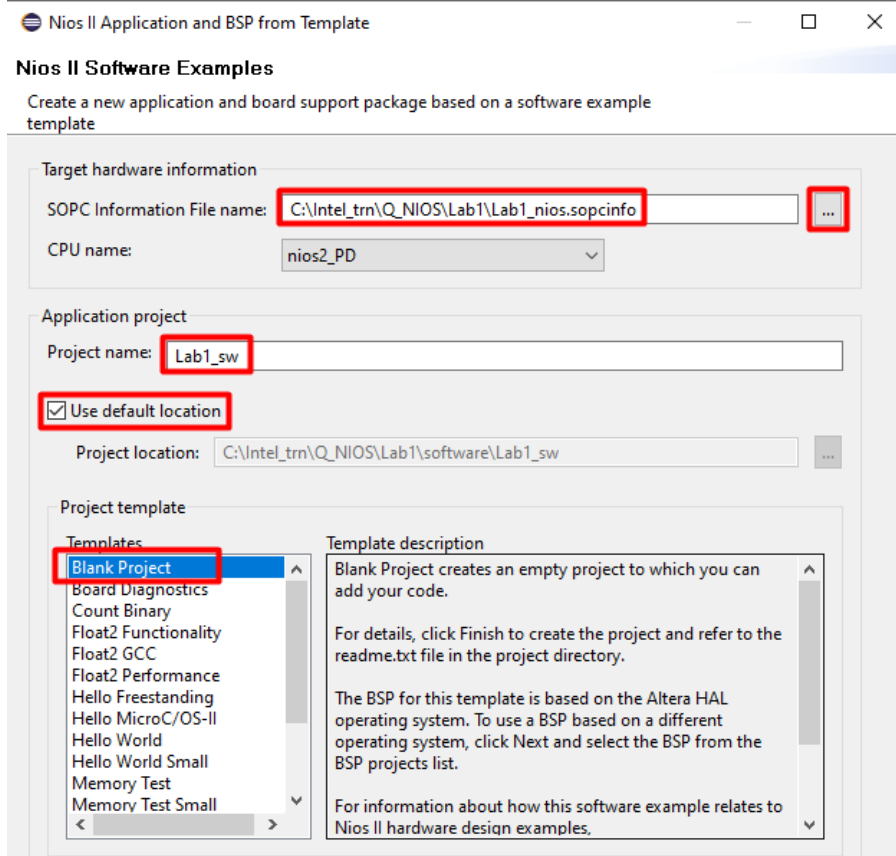


- ✓ Назначьте рабочую область для данной лабораторной
- ✓ Нажмите кнопку OK.

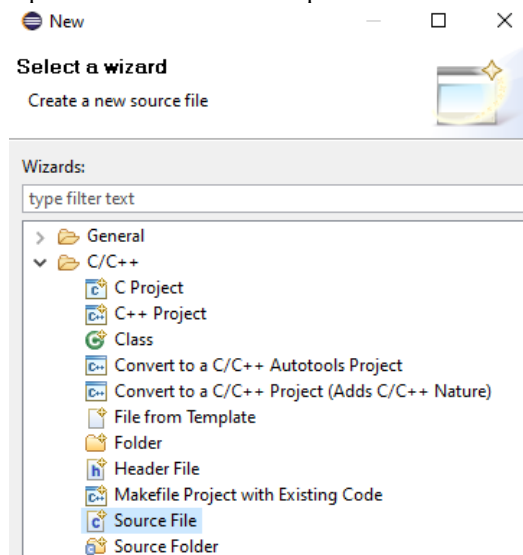
21. Выполните команду File=>New=>NIOS II Application and BSP from Template.



- ✓ Будет запущен помощник создания нового проекта – New Project Wizard
- ✓ В окне помощника введите:

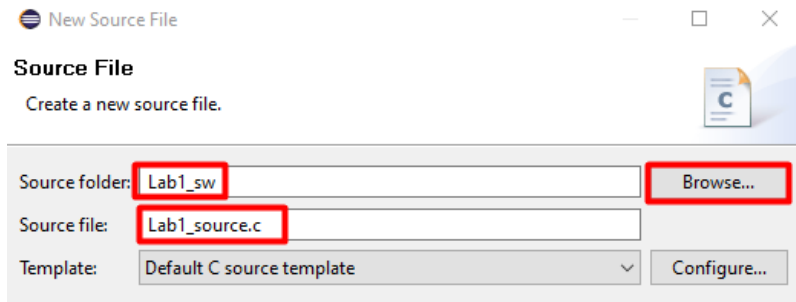


1. В разделе Target Hardware Information с помощью браузера найдите в рабочей папке и укажите файл lab1\_nios.sopcinfo – файл с описанием созданной системы.
  2. В разделе Application Project введите название проекта – Lab1\_sw
  3. В разделе Select Project Template выберите Blank Project
  4. Нажмите 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:





- ✓ Найдите и укажите папку Lab1\_sw,
- ✓ введите название файла: Lab1\_source.c;
- ✓ Выберите Template => Default C source template.
- ✓ Нажмите кнопку Finish.

*Будет создан и открыт в текстовом редакторе новый файл.*

22. Введите текст программы на языке Си:

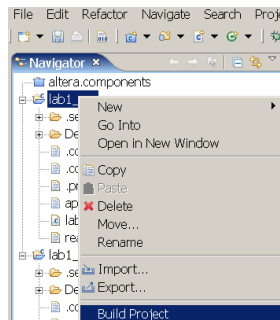
```
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include <unistd.h>

int main(void)
{
    int sw;
    int count = 255;

    while( 1 )
    {
        usleep (500000);
        sw = IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE); /* read sw[0] value */
        if (sw == 0x1) count++; /* Continue 0-ff counting loop. */
        else count--; /* Continue ff-0 counting loop. */
        IOWR_ALTERA_AVALON_PIO_DATA( PIO_LED_BASE, ~count );
    }
    return 0;
}
```

- ✓ Сохраните изменения.

23. Выберите папку Lab1\_sw, нажмите правую клавишу мыши и укажите команду Build Project. Будет запущен компилятор.



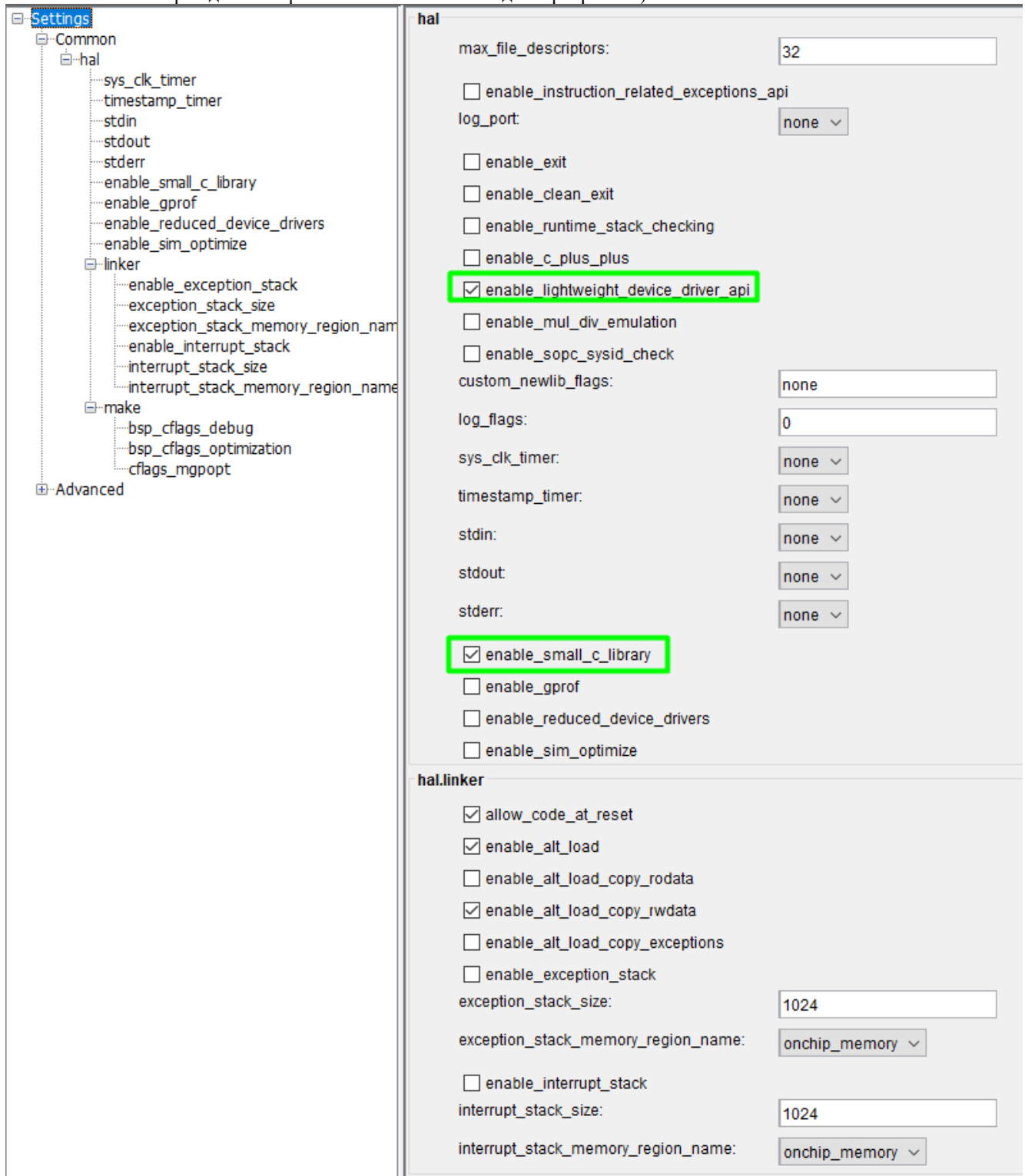
24. При успешном завершении процесса, в окне Console появится сообщение
- ```
nios2-elf-insert Lab1_sw.elf --thread_model hal --cpu_name nios2_PD --qsys true --simulatic
Info: (Lab1_sw.elf) 4568 Bytes program size (code + initialized data).
Info: 10 KBytes free for stack + heap.
Info: Creating Lab1_sw.objdump
nios2-elf-objdump --disassemble --syms --all-header --source Lab1_sw.elf >Lab1_sw.objdump
[Lab1_sw build complete]
```
- 09:43:17 Build Finished (took 8s.750ms)

*Обратите внимание на размер памяти, оставшейся свободной (из ОЗУ 16 Кбайт , указанных при создании платформы, программой и данными инициализации занято 4568 Байт, свободно 10кБайт).*

25. Уменьшение размера программы:

- ✓ Выберите папку Lab1\_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 так, как показано на рисунке, приведенном ниже (это поможет сократить объем порождаемого файла с исполняемым кодом программы).



- ✓ Нажмите кнопку Generate.  
✓ Затем кнопку Exit.

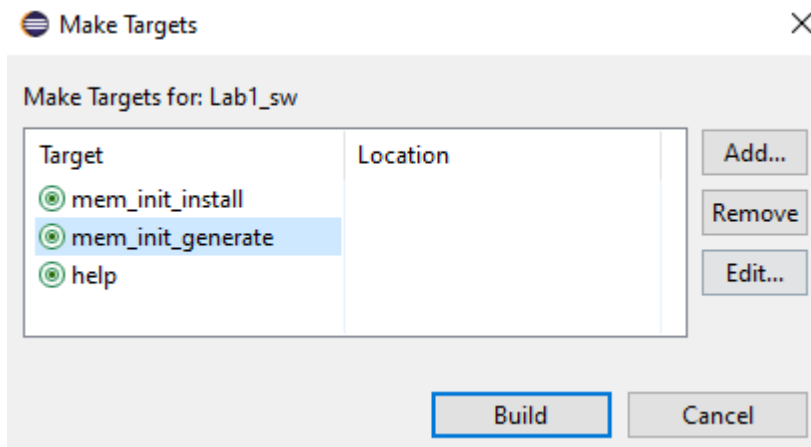
26. Выберите папку Lab1\_sw, нажмите правую клавишу мыши и укажите команду Build Project. Будет запущен компилятор.
27. При успешном завершении процесса, в окне Console появится сообщение
- ```
nios2-elf-insert Lab1_sw.elf --thread model hal --cpu name nios2 PD --qsys true --simulation
Info: (Lab1_sw.elf) 1584 Bytes program size (code + initialized data).
Info: 14 KBytes free for stack + heap.
Info: Creating Lab1_sw.objdump
nios2-elf-objdump --disassemble --syms --all-header --source Lab1_sw.elf >Lab1_sw.objdump
[Lab1_sw build complete]
```

09:53:51 Build Finished (took 7s.882ms)

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

## 28. Создание hex файла для инициализации памяти FPGA:

- ✓ Выберите папку Lab1\_sw, нажмите правую клавишу мыши и укажите команду Make Targets => Build.
- ✓ В появившемся окне выберите mem\_init\_generate и нажмите Build. Будет запущена генерация файла инициализации памяти.



## 29. При успешном завершении процесса, в окне Console появится сообщение, в котором указано:

- ✓ Размер и базовый адрес памяти, указанные при создании системы в пакете PD (убедитесь в том, что эти адреса соответствуют друг другу)
- ✓ Папка, в которой создан файл инициализации памяти
- ✓ qip файл, который следует подключить к проекту в пакете QP

```
[BSP build complete]
Post-processing to create mem_init/onchip_mem.hex...
elf2hex Lab1_sw.elf 0x00004000 0x00007fff --width=32 --little-endian-mem --create-lanes=0 mem_init/onchip_mem.hex
Post-processing to create mem_init/hdl_sim/onchip_mem.dat...
elf2dat --infile=Lab1_sw.elf --outfile=mem_init/hdl_sim/onchip_mem.dat \
--base=0x00004000 --end=0x00007fff --width=32 \
--little-endian-mem --create-lanes=0
Post-processing to create mem_init/hdl_sim/onchip_mem.sym...
nios2-elf-nm -n Lab1_sw.elf > mem_init/hdl_sim/onchip_mem.sym
Post-processing to create mem_init/meminit.spd...
Post-processing to create mem_init/meminit.qip...
```

09:58:24 Build Finished (took 1s.742ms)

*Компиляция программы закончена успешно.  
Создан onchip\_mem.hex файл для инициализации модуля памяти программ процессора.*

## Часть 4 – Полная компиляция проекта

### 1. В пакете QP подключите к проекту файл с описанием файла инициализации памяти, созданного в NiosII IDE:

- ✓ Выполните команду Project=>Add/Remove Files in Project
- ✓ В появившемся окне, в разделе File Name выберите (с помощью браузера) файл ...  
C:\Intel\_trn\Q\_NIOS\Lab1\software\Lab1\_sw\mem\_init\meminit.qip
- ✓ Нажмите кнопку Add. Затем кнопку OK.

### 2. С помощью Timing Analyzer или в текстовом редакторе создайте файл (**Lab1.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
#####

set_clock_uncertainty -rise_from [get_clocks {clock}] -rise_to [get_clocks {clock}] 0.020
set_clock_uncertainty -rise_from [get_clocks {clock}] -fall_to [get_clocks {clock}] 0.020
set_clock_uncertainty -fall_from [get_clocks {clock}] -rise_to [get_clocks {clock}] 0.020
set_clock_uncertainty -fall_from [get_clocks {clock}] -fall_to [get_clocks {clock}] 0.020

#####
# 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}]

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

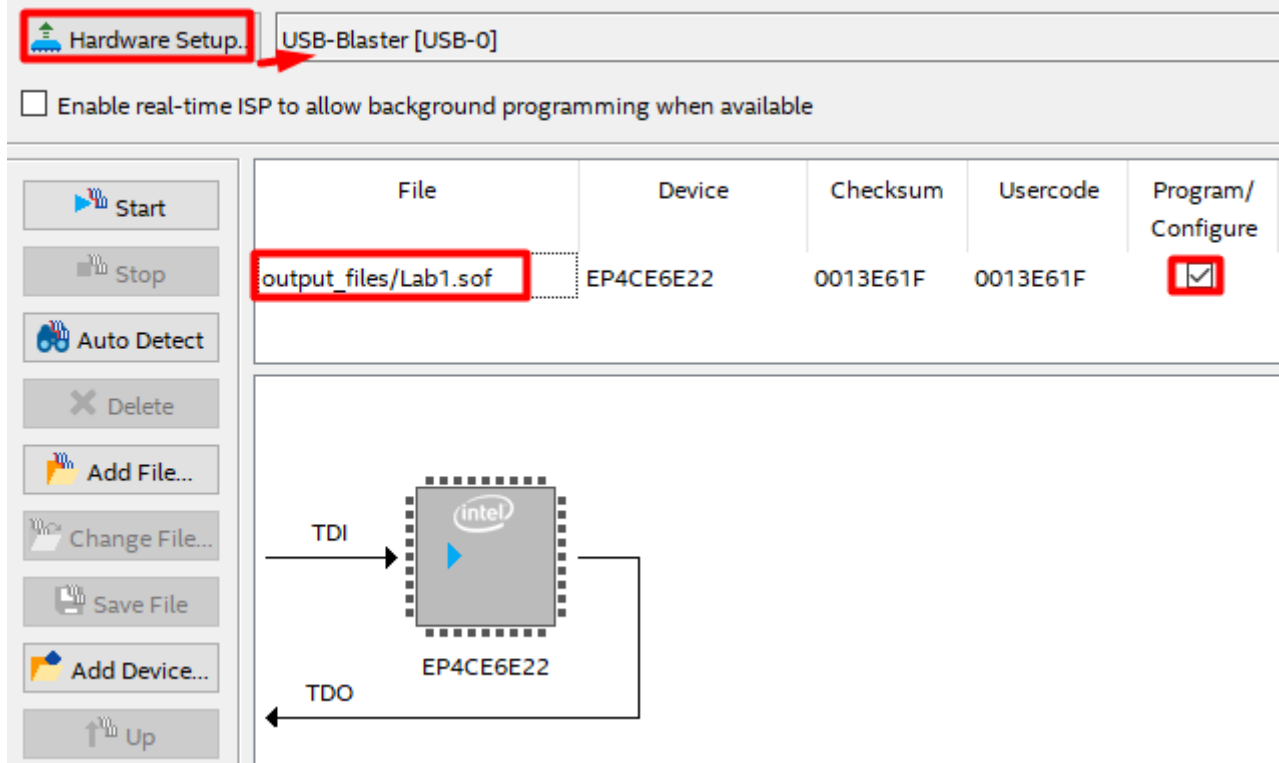
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[0]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[1]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[2]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 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]}]

```

3. Подключите файл Lab1.sdc к проекту.
  - a. Выполните команду Assignment=>Settings
  - b. В разделе Timing Analyzer добавьте файл Lab1.sdc к проекту.
4. В окне менеджера пакета QP, с помощью команды **Processing => Start Compilation** осуществите полную компиляцию проекта.

## Часть 5 – Конфигурирование СБИС

1. Подключите плату miniDilabCIV к ПК. Включите питание платы.
2. Включите питание платы.
3. Выполните команду **Tools=> Programmer**
4. Откроется окно управления конфигурированием СБИС.
  - a. Установите средство конфигурирования FPGA
  - b. Выберите файл для конфигурирования
  - c. Включите опцию **Program/Configure**
  - d. Нажмите кнопку **Start**.
5. В окне Progress будет отображаться статус процедуры программирования.



6. Когда СБИС будет запрограммирована на плате загорится зеленый светодиод.



7. Проверьте работу проекта

- а. Светодиоды LED8...LED1 на плате начнут переключаться, отображая в двоичном коде 8-разрядное число, изменяющееся
  - i. от 0 до 255 если переключатель sw[0] в положении 1
  - ii. от 255 до 0 если переключатель sw[0] в положении 0

## Часть 6 – Дополнительные задания

## I. Задание 1

- Измените текст программы так, чтобы использовать указатели (абсолютные адреса элементов PIO\_SW и PIO\_LED были заданы при создании системы в пакете PD

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		<b>clk</b>	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		
<input checked="" type="checkbox"/>		<b>onchip_memory</b>	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"/>		<b>nios2_PD</b>	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
		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]		IRQ 31
<input checked="" type="checkbox"/>		<b>pio_LED</b>	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]	0x8010	0x801f
		external_connection	Conduit				
<input checked="" type="checkbox"/>		<b>pio_SW</b>	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]	0x8000	0x800f
		external_connection	Conduit				
				sw			

- Пример исходного кода приведен на рисунке (кроме использования указателей, увеличена частота переключения светодиодов – уменьшено значение функции usleep; изменено начальное значение счетчика)

```

1 // #include "system.h"
2 // #include "altera_avalon_pio_regs.h"
3 // #include <unistd.h>
4
5 // int main(void)
6 // {
7 //     int sw ;
8 //     int count = 255;
9 //     while( 1 )
10 //     {
11 //         usleep (500000);
12 //         sw = IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE); /* read sw[0] value */
13 //         if (sw == 0x1) count++; /* Continue 0-ff counting loop. */
14 //         else
15 //             count--; /* Continue ff-0 counting loop. */
16 //         IOWR_ALTERA_AVALON_PIO_DATA( PIO_LED_BASE, ~count ); /* write value to Led[7:0] */
17 //     }
18 //     return 0;
19 // }
20
21 #include <unistd.h>
22
23 int main(void)
24 {
25     int *psw = (int*) 0x8000;
26     int *pled = (int*) 0x8010;
27     int count = 64;
28
29     while( 1 )
30     {
31         usleep (100000);
32         if (*psw == 0x1) count++; /* Continue 0-ff counting loop. */
33         else
34             count--; /* Continue ff-0 counting loop. */
35         *pled = ~count;
36     }
37     return 0;
38 }
39

```

- Проверьте реализацию на плате.

## II. Задание 2

1. Измените (в структуре системы) разрядность SW с одного разряда до 8 разрядов (и подключите к переключателям SW[7:0]).
2. Создайте текст программы на языке Си, обеспечивающий счет на сложение по модулю, заданному переключателями SW[7:0].
3. Пример кода приведен на рисунке (это только пример, адреса, заданные указателями у Вас будут другими)

```
Lab1s_source.c
1 #include <unistd.h>
2
3 int main(void)
4 {
5     char *psw = (char*)    0x2000;
6     char *pled = (char*)    0x1000;
7     char count = 64;
8
9     while( 1 )
10    {
11        usleep (300000);
12
13        if ( ( (*psw)!=0x00 ) & ( ( (*psw)-1) > count) )
14            count++;    /* Continue 0-SW[7:0] counting loop. */
15        else
16            count = 0; /* start counting loop from 0 */
17
18        *pled = ~count;
19    }
20    return 0;
21 }
```

4. Проверьте реализацию на плате.

Упражнение завершено.