

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа компьютерных технологий и информационных систем

Отчёт по лабораторной NIOSII_2

Дисциплина: Автоматизация проектирования дискретных устройств (на
английском языке)

Выполнил студент гр. 5130901/10101 _____ М.Т. Непомнящий
(подпись)

Руководитель _____ А.А. Федотов
(подпись)

Санкт-Петербург
2024

Оглавление

1.	Задание.....	4
1.1.	Цель работы	4
1.2.	Структура проекта.....	4
2.	Ход работы	5
2.1.	Создание проекта.....	5
	Начало работы в PD	5
2.2.	Настройка сигналов.....	5
	Изменение настроек Nios II Processor	6
	Адресация системы	6
2.3.	Анализ полученной системы.....	7
	Анализ предустановок	7
	Проверка блока.....	8
	Проверка отсутствия проблемных подключений	8
	Анализ с помощью Schematic	9
	Генерация системы	9
2.4.	Подключение файлов к проекту.....	11
2.5.	Анализ проекта	11
	RTL Viewer	11
	Назначение выходов проекта.....	12
2.6.	Создание программной части проекта	12
	Создание файла Source	13
	Компиляция проекта в Eclipse	14
	Полная компиляция проекта в QP	15
2.7.	Реализация на плате	16
	Конфигурирование FPGA	16
	Загрузка ПО	16
	Запуск на плате.....	16
	Отладка ПО.....	17
	Повторный запуск на плате.....	17
3.	Дополнительное задание.....	20
3.1.	Создание программного файла Lab2_extra_task2_source.c.....	20
3.2.	Реализация на плате	20
4.	Вывод	21

Список иллюстраций

Рис. 1 – Структура проекта.....	4
Рис. 2 – Создание проекта.....	5
Рис. 3 – Исходное окно PD	5
Рис. 4 – Настройка модуля nios2_PD.....	6
Рис. 5 – Подключение модуля nios2_PD	6
Рис. 6 – Окно Address Map после выполнения автоматической адресации	6
Рис. 7 – Внешний вид созданной системы.....	7
Рис. 8 – Предустановки системы	7
Рис. 9 – Окно Messages с предупреждением.....	7
Рис. 10 – Символ системы	8
Рис. 11 – Анализ проблемных подключений.....	8
Рис. 12 – Show System with QSYS Interconnect.....	9
Рис. 13 – Schematic	9
Рис. 14 – Предустановки окна Generation (по умолчанию).....	10
Рис. 15 – Проверка генерации HDL	10
Рис. 16 – Подключение файла .qip к проекту	11
Рис. 17 – Синтаксис файла Lab1.sv	11
Рис. 18 – Схема проекта в RTL Viewer.....	11
Рис. 19 – Назначение выводов платы средствами Pin Planner	12
Рис. 20 – Unused Pins.....	12
Рис. 21 – Создание проекта в Eclipse.....	13
Рис. 22 – Пред настройки Source File'a	13
Рис. 23 – Синтаксис файла Lab2_source.c	13
Рис. 24 – Build Project.....	14
Рис. 25 – Настройка BSP Editor.....	14
Рис. 26 – Build Project после изменения настроек BSP Editor.....	14
Рис. 27 – SDC файл.....	15
Рис. 28 – Добавление SDC файла.....	15
Рис. 29 – Временные характеристики устройства.....	15
Рис. 30 – Загрузка ПО проекта средствами Eclipse.....	16
Рис. 31 – Refresh Connections.....	16
Рис. 32 – Режим отладки NiosII Debug	17
Рис. 33 – Установка точек прерывания	18
Рис. 34 – Установка указателя &led.....	18
Рис. 35 – Закладка Variables (1).....	18
Рис. 36 – Закладка Memory (1)	19
Рис. 37 – Закладка Variables (2).....	19
Рис. 38 – Закладка Memory	19
Рис. 39 – Изменённый исходный файл.....	20

1. Задание

1.1. Цель работы

Расширить знакомство с возможностями по реализации проектов на базе процессора NIOSII:

- ✓ 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

1.2. Структура проекта

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

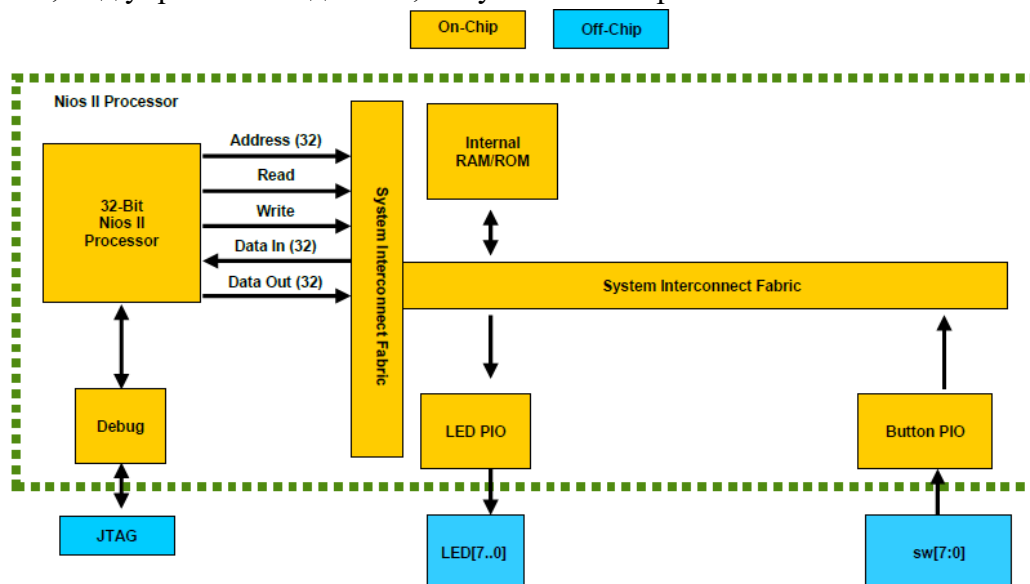


Рис. 1 – Структура проекта

Устройство, которое содержит master и 3 slave: 2 модуля my_slave и 1 модуль my_Dslave (default slave)

Master получает некоторые данные через Conduit, через 8-разрядный интерфейс мастер осуществляет адресный доступ к одному из slave'ов, настраивает соответственно slave's, либо что-то в них записывает, каждый из slave'ов имеет в себе Conduit, который помогает посмотреть на выводе slave'a то, что мы туда записали из мастера.

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).

2. Ход работы

2.1. Создание проекта

Создадим проект, указав параметры, представленные на Рис. 2 ниже:

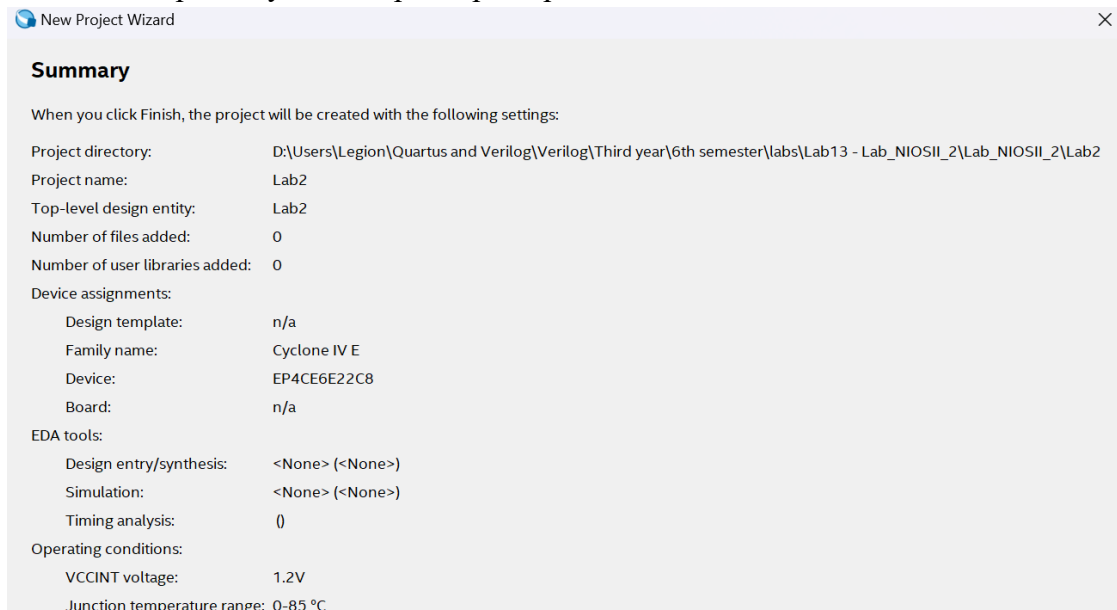


Рис. 2 – Создание проекта

Начало работы в PD

Откроем PD и сохраним систему:

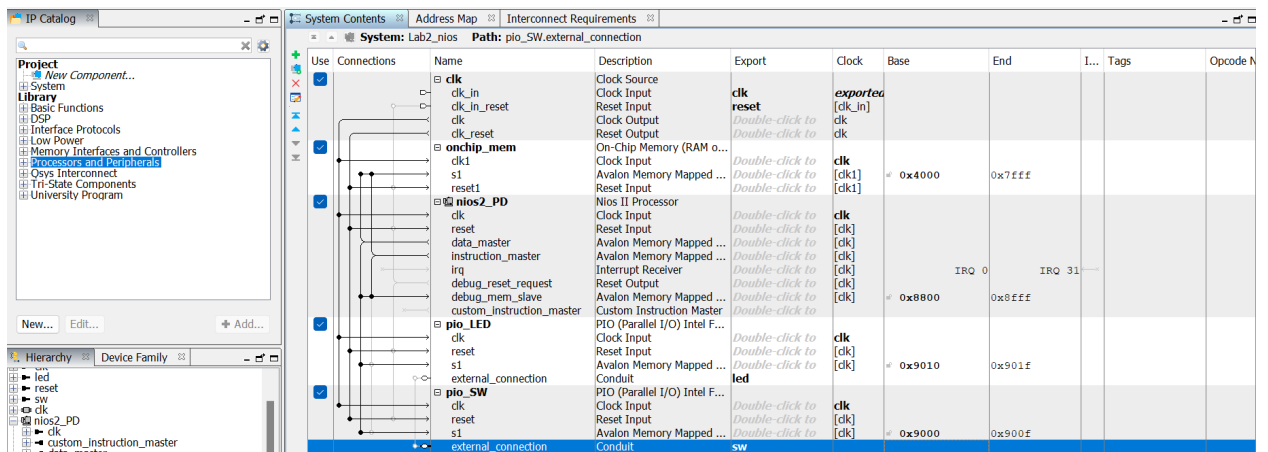


Рис. 3 – Исходное окно PD

2.2. Настройка сигналов

Изменение настроек Nios II Processor

В параметрах модуля откроем страницу JTAG DEBUG, где поставим галочку у режима Include JTAG Debug

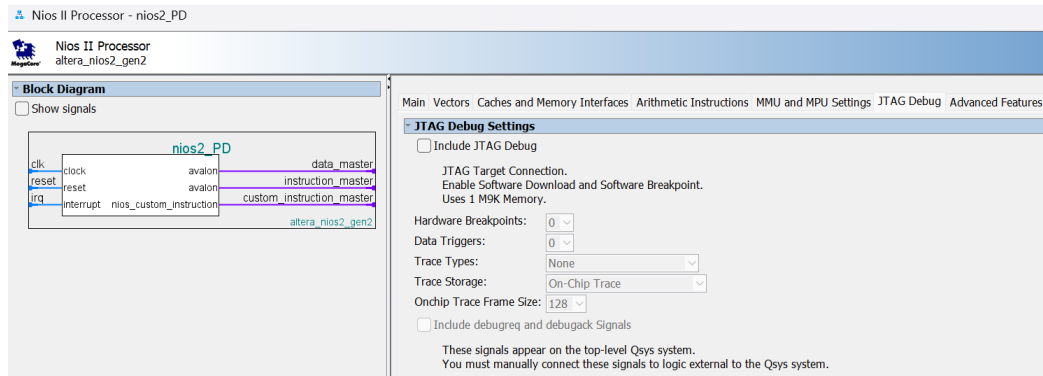


Рис. 4 – Настройка модуля nios2_PD

Добавим подключения к данному модулю. Для этого соединим data_master и instruction_master компонента nios2_PD с входом debug_mem_slave компонента nios2_PD:

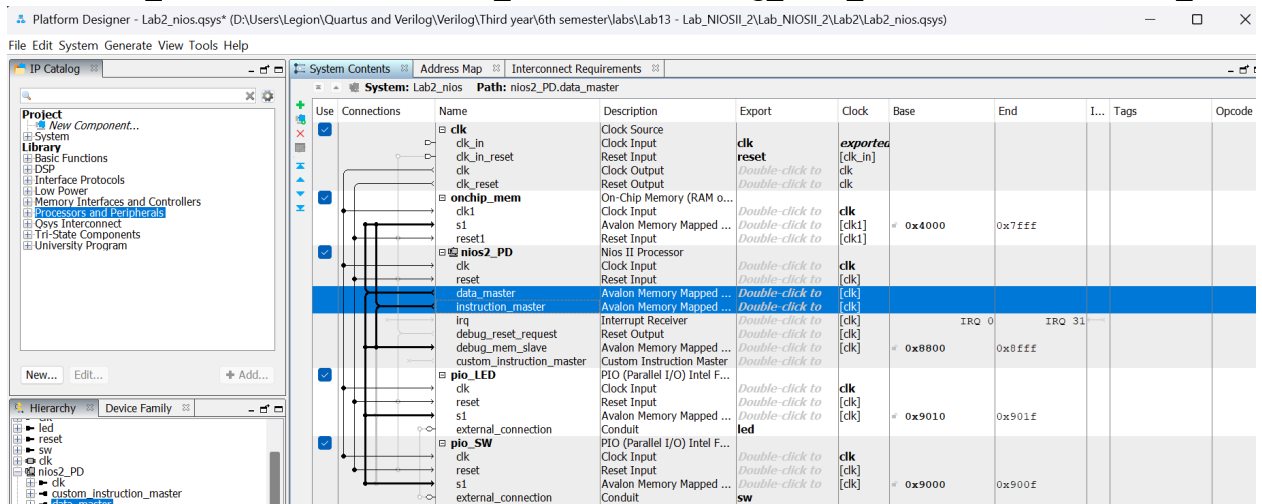


Рис. 5 – Подключение модуля nios2_PD

Адресация системы

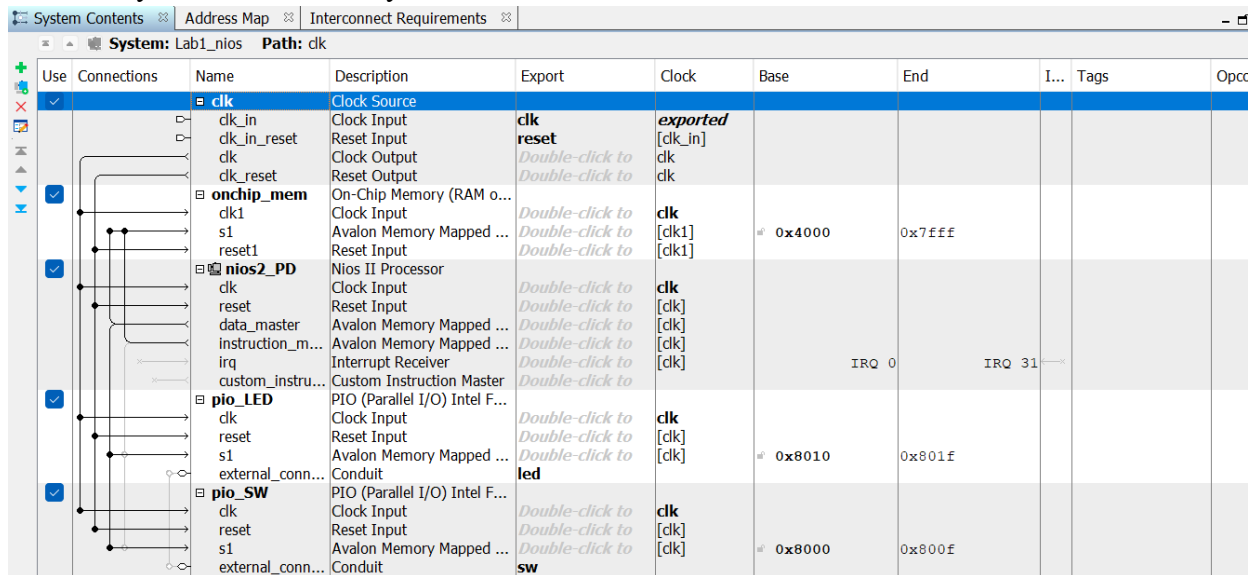
Выполним System → Assign base Addresses. После автоматической адресации модулей окно Address Map будет выглядеть следующим образом:

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

Рис. 6 – Окно Address Map после выполнения автоматической адресации

2.3. Анализ полученной системы

Полученная система будет выглядеть так, как показано на ниже:

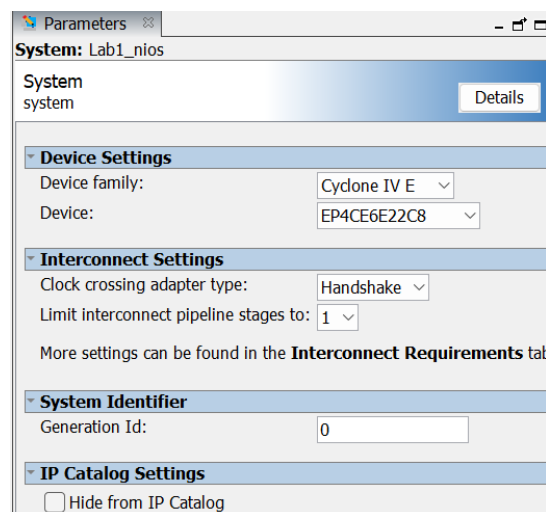


Use	Connections	Name	Description	Export	Clock	Base	End	I...	Tags	Opcc
✓		clk	Clock Source							
		clk_in	Clock Input	clk	exported					
		clk_in_reset	Reset Input	reset	[clk_in]					
		clk	Clock Output	Double-click to	clk					
		clk_reset	Reset Output	Double-click to	clk					
✓		onchip_mem	On-Chip Memory (RAM o...							
		clk1	Clock Input	Double-click to	clk	# 0x4000	0x7fff			
		s1	Avalon Memory Mapped ...	Double-click to	[clk1]					
		reset1	Reset Input	Double-click to	[clk1]					
✓		nios2_PD	Nios II Processor							
		clk	Clock Input	Double-click to	clk					
		reset	Reset Input	Double-click to	[clk]					
		data_master	Avalon Memory Mapped ...	Double-click to	[clk]					
		instruction_m...	Avalon Memory Mapped ...	Double-click to	[clk]					
		irq	Interrupt Receiver	Double-click to	[clk]			IRQ 0	IRQ 31	
		custom_instru...	Custom Instruction Master	Double-click to	[clk]					
✓		pio_LED	PIO (Parallel I/O) Intel F...							
		clk	Clock Input	Double-click to	clk	# 0x8010	0x801f			
		reset	Reset Input	Double-click to	[clk]					
		s1	Avalon Memory Mapped ...	Double-click to	[clk]					
		external_conn...	Conduit	Double-click to	led					
✓		pio_SW	PIO (Parallel I/O) Intel F...							
		clk	Clock Input	Double-click to	clk	# 0x8000	0x800f			
		reset	Reset Input	Double-click to	[clk]					
		s1	Avalon Memory Mapped ...	Double-click to	[clk]					
		external_conn...	Conduit	Double-click to	sw					

Рис. 7 – Внешний вид созданной системы

Анализ предустановок

Проверим, что предустановки для полученной системы указаны так, как показано на рис. 17 ниже:



Parameters

System: Lab1_nios

System system Details

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

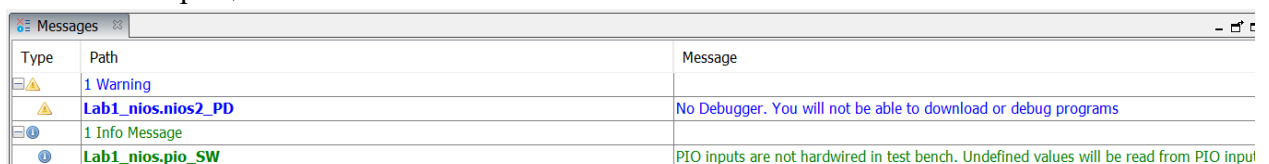
Generation Id: 0

IP Catalog Settings

☐ Hide from IP Catalog

Рис. 8 – Предустановки системы

В окне Messages есть только 1 предупреждение, связанное с тем, что не подключён JTAG Debug модуль, однако это было сделано намеренно, поэтому на это предупреждение можем не обращать внимание:



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

Рис. 9 – Окно Messages с предупреждением

Проверка блока

Выполним View → Block Symbol и убедимся в том, что символ системы построен правильно:

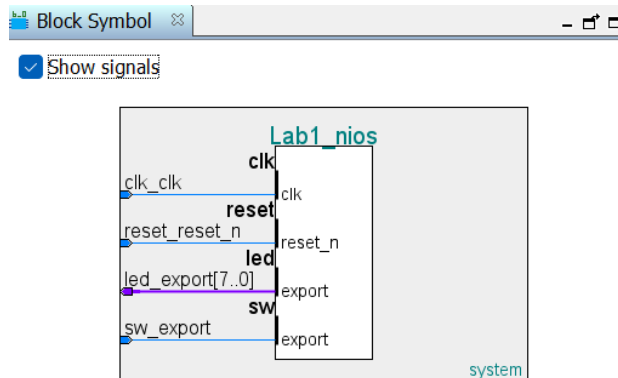


Рис. 10 – Символ системы

Проверка отсутствия проблемных подключений

Выполним View → Clock domains Beta, выберем режим отображения Reset, затем Clock. Заметим, что проблемных подключений ни в одном из случаев не выявлено:

System Contents			Address Map		Interconnect Requirements				
System: Lab2_nios									
Use	Connections	Name	Description	Export	Clock	Base	End	I... Tags	
<input checked="" type="checkbox"/>		clk	Clock Source	clk	exported [clk_in] clk	# 0x4000	0x7fff		
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk					
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset	Double-click to Double-click to	clk [clk1] [clk1]			
<input checked="" type="checkbox"/>		clk	Clock Output	clk	Double-click to Double-click to	clk [clk] [clk]			
<input checked="" type="checkbox"/>		clk_reset	Reset Output	reset	Double-click to Double-click to	clk [clk] [clk]			
<input checked="" type="checkbox"/>		onchip_mem	On-Chip Memory (RAM o...	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		clk1	Clock Input	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped ...	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		reset1	Reset Input	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		nios2_PD	Nios II Processor	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		clk	Clock Input	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped ...	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped ...	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		irq	Interrupt Receiver	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		debug_reset_request	Reset Output	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		debug_mem_slave	Avalon Memory Mapped ...	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		custom_instruction_master	Custom Instruction Master	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		pio_LED	PIO (Parallel I/O) Intel F...	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>		clk	Clock Input	Double-click to Double-click to	clk [clk] [clk]				
<input checked="" type="checkbox"/>	reset	Reset Input	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped ...	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	external_connection	Conduit	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	pio_SW	PIO (Parallel I/O) Intel F...	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	clk	Clock Input	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	reset	Reset Input	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped ...	Double-click to Double-click to	clk [clk] [clk]					
<input checked="" type="checkbox"/>	external_connection	Conduit	Double-click to Double-click to	clk [clk] [clk]					

<

Рис. 11 – Анализ проблемных подключений

Выполним команду System → Show System with PD Interconnect (Show System with QSYS Interconnect). Проверим, был добавлен только модуль mm_interconnect_0.

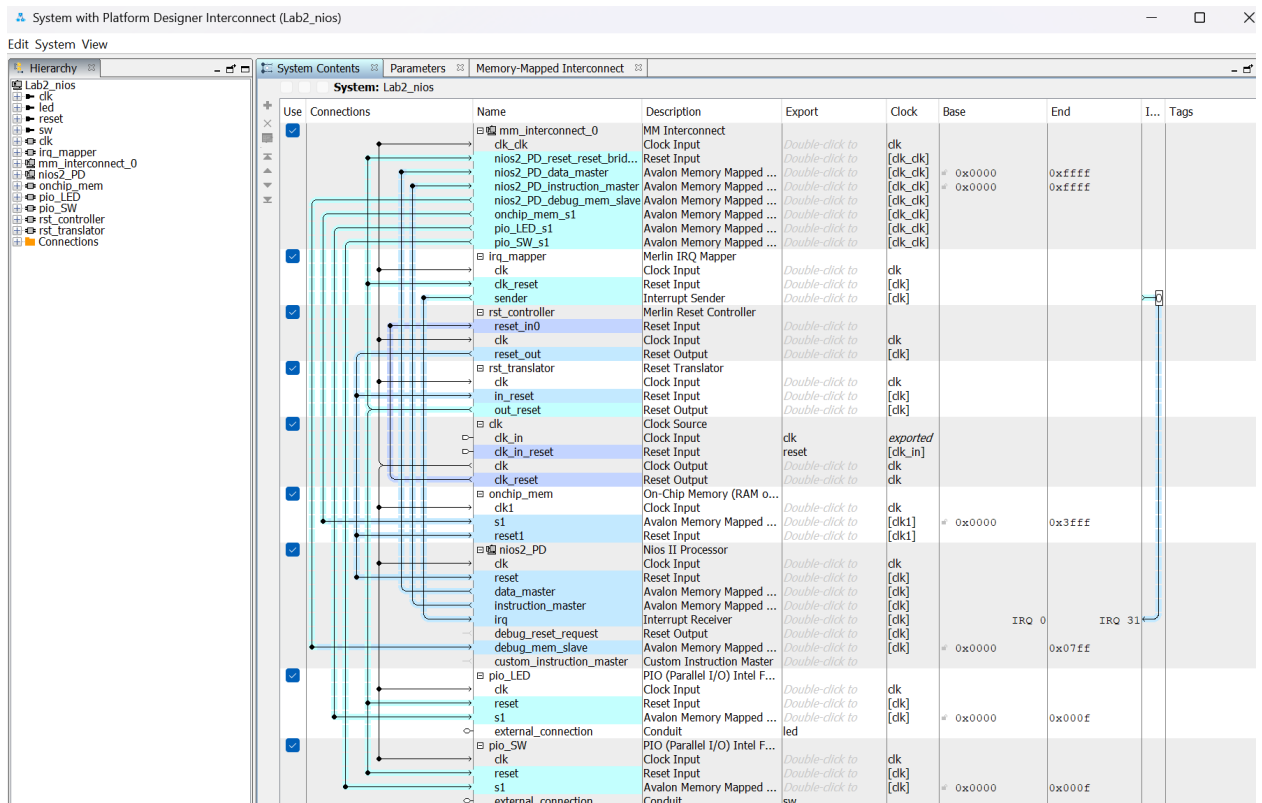


Рис. 12 – Show System with QSYS Interconnect

Анализ с помощью Schematic

Выполним View → Schematic, в качестве фильтра введём in и убедимся в том, что система синхронизации и каналы ST системы подключены верно:

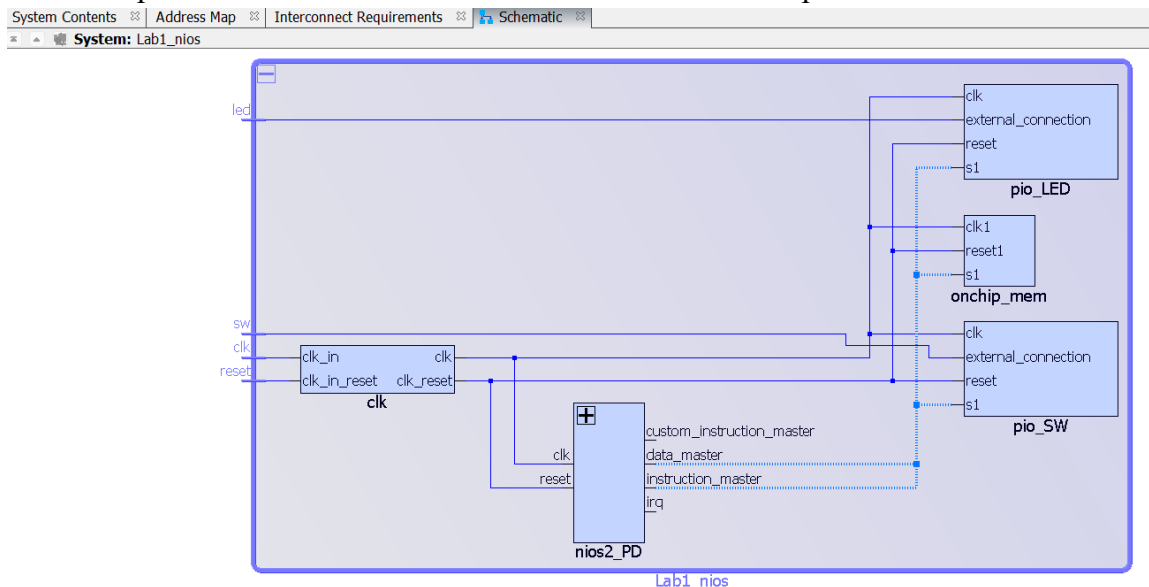


Рис. 13 – Schematic

Генерация системы

Выполним PD → Generate HDL и укажем следующие предустановки для генерации:

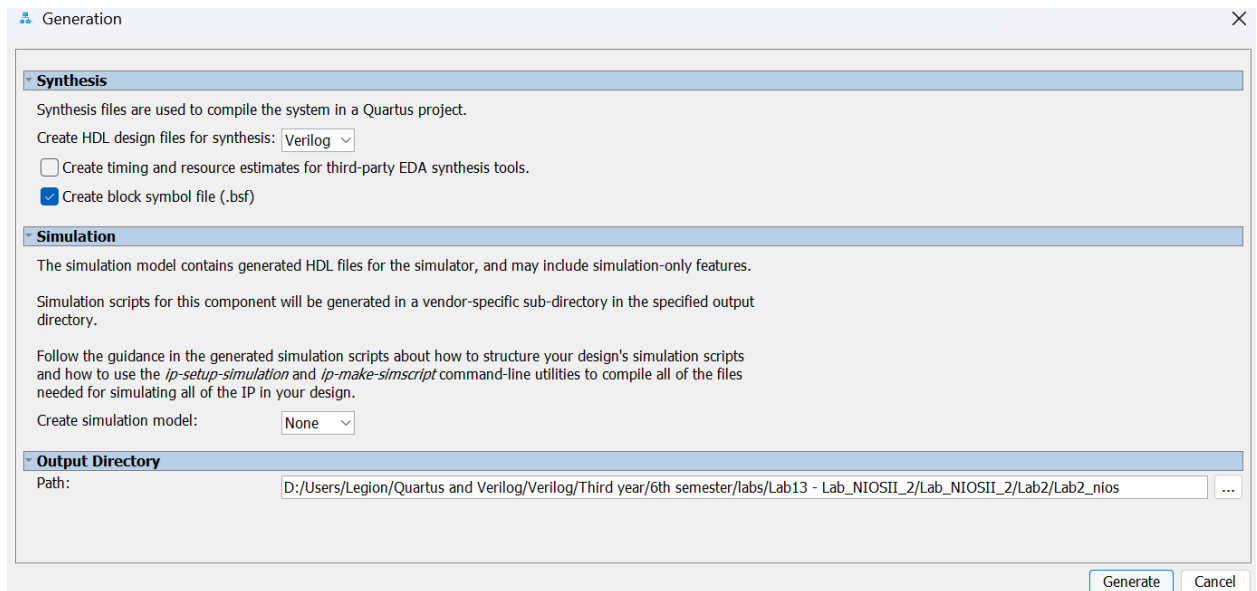


Рис. 14 – Предустановки окна Generation (по умолчанию)

Удостоверимся в том, что среди предупреждений есть только 1 пункт, связанный с JTAG, про сто говорилось выше:

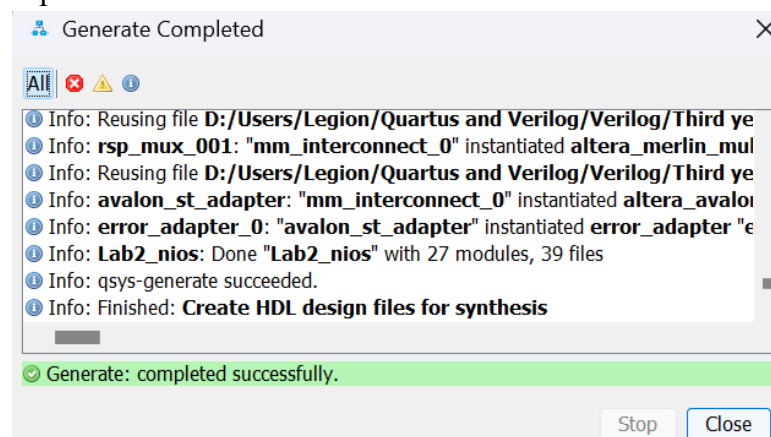


Рис. 15 – Проверка генерации HDL

2.4. Подключение файлов к проекту

Подключим файл .qip только что созданной системы к проекту в Quartus

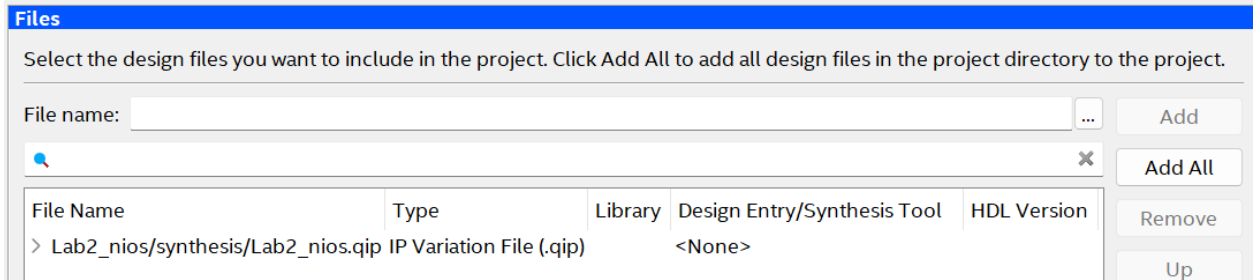


Рис. 16 – Подключение файла .qir к проекту

Создадим файл верхнего уровня `Lab1.sv`, синтаксис которого приведён ниже:

```

1  module lab2 (
2      input bit clk,
3      input bit [7:0] sw,
4      input bit pbb,
5      output bit [7:0] led
6  );
7
8  Lab2_nios u0 (
9      .clk_clk(clk),
10     .reset_reset_n(pbb),
11     .led_export(led),
12     .sw_export(sw)
13 );
14
15 endmodule
16

```

Рис. 17 – Синтаксис файла Lab1.sv

2.5. Анализ проекта

RTL Viewer

Выполним анализ и синтез проекта средствами QP и убедимся в правильности схемы средствами RTL Viewer:

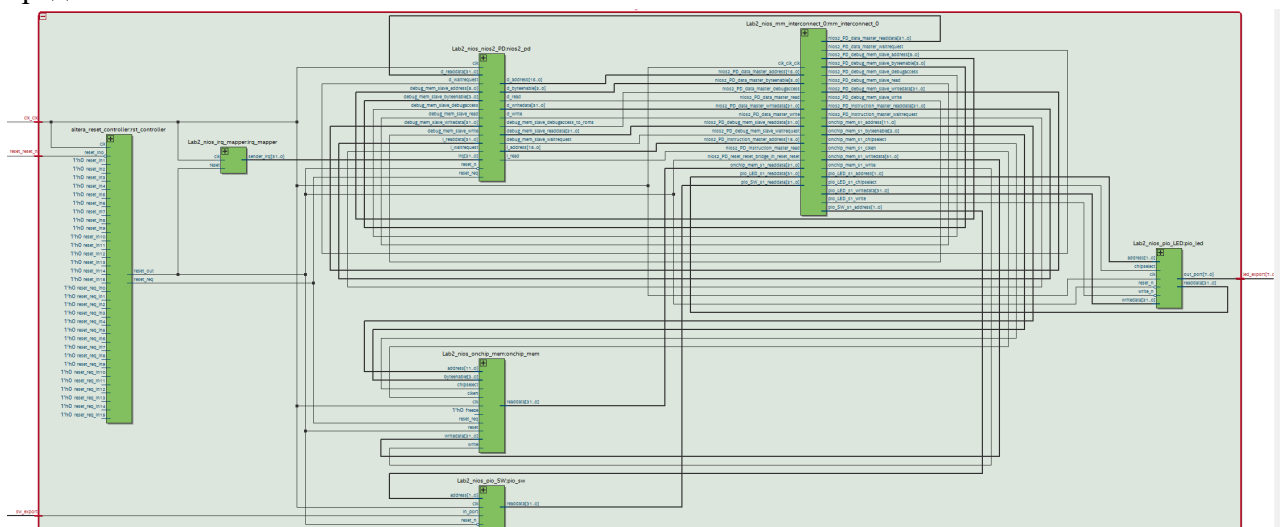


Рис. 18 – Схема проекта в RTL Viewer

Можем увидеть, что полученная в RTL Viewer схема совпадает с той, что была задана по условию (в зелёном блоке отображается тот фрагмент системы, который был создан средствами PD).

Назначение выходов проекта

Откроем редактор назначения выводов (Pin Planner): Assignment → Pin Planner. Назначим выводы на плату так, как показано на ниже:

Named: * <div><div></div><div>Edit: x</div><div></div></div>										
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Current Strength	Slew Rate	Reserved	Differential Pair	Strict Preservation
clk	Input	PIN_23	1	B1_NO	3.3-V LVTTTL	8mA (default)				
led[7]	Output	PIN_65	4	B4_NO	2.5 V	8ma	2 (default)			
led[6]	Output	PIN_66	4	B4_NO	2.5 V	8ma	2 (default)			
led[5]	Output	PIN_67	4	B4_NO	2.5 V	8ma	2 (default)			
led[4]	Output	PIN_68	4	B4_NO	2.5 V	8ma	2 (default)			
led[3]	Output	PIN_69	4	B4_NO	2.5 V	8ma	2 (default)			
led[2]	Output	PIN_70	4	B4_NO	2.5 V	8ma	2 (default)			
led[1]	Output	PIN_71	4	B4_NO	2.5 V	8ma	2 (default)			
led[0]	Output	PIN_72	4	B4_NO	2.5 V	8ma	2 (default)			
pbb	Input	PIN_58	4	B4_NO	2.5 V	8mA (default)				
sw[7]	Input	PIN_88	5	B5_NO	3.3-V LVTTTL	8mA (default)				
sw[6]	Input	PIN_89	5	B5_NO	3.3-V LVTTTL	8mA (default)				
sw[5]	Input	PIN_90	6	B6_NO	3.3-V LVTTTL	8mA (default)				
sw[4]	Input	PIN_91	6	B6_NO	3.0-V LVTTTL	8mA (default)				
sw[3]	Input	PIN_49	3	B3_NO	3.0-V LVTTTL	8mA (default)				
sw[2]	Input	PIN_46	3	B3_NO	3.3-V LVTTTL	8mA (default)				
sw[1]	Input	PIN_25	2	B2_NO	3.3-V LVTTTL	8mA (default)				
sw[0]	Input	PIN_24	2	B2_NO	3.3-V LVTTTL	8mA (default)				
<<ne_de>>										

Рис. 19 – Назначение выводов платы средствами Pin Planner

Выполним Assignment → Device → Device and Pin Options → Unused Pins, в появившемся окне установим опцию As input tri-stated with weak pull-up:

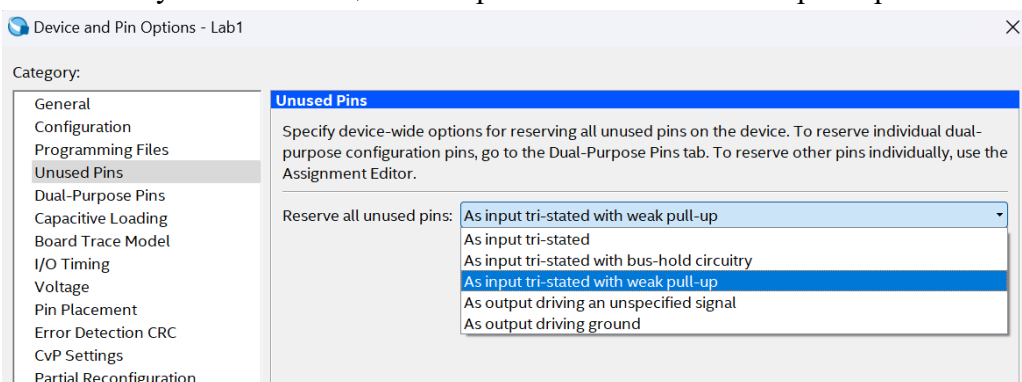


Рис. 20 – Unused Pins

2.6. Создание программной части проекта

Создадим оболочку Nios II SBT (Software Build Tools) средствами Eclipse.

Укажем файл с описанием программы lab1_nios.sprocinfo. В качестве названия проекта укажем Lab1_sw.

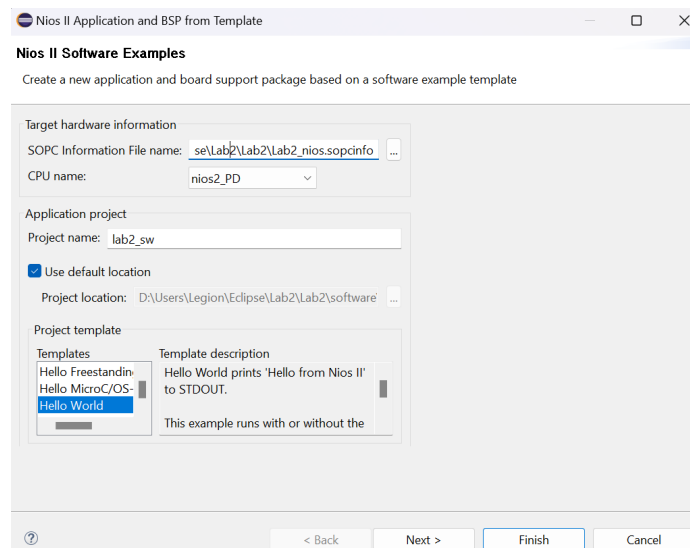


Рис. 21 – Создание проекта в Eclipse

Создание файла Source

Создадим Source File:

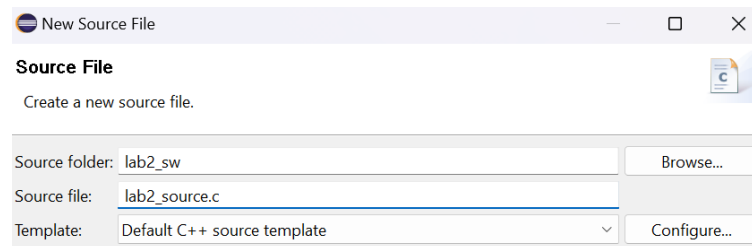


Рис. 22 – Пред настройки Source File'a

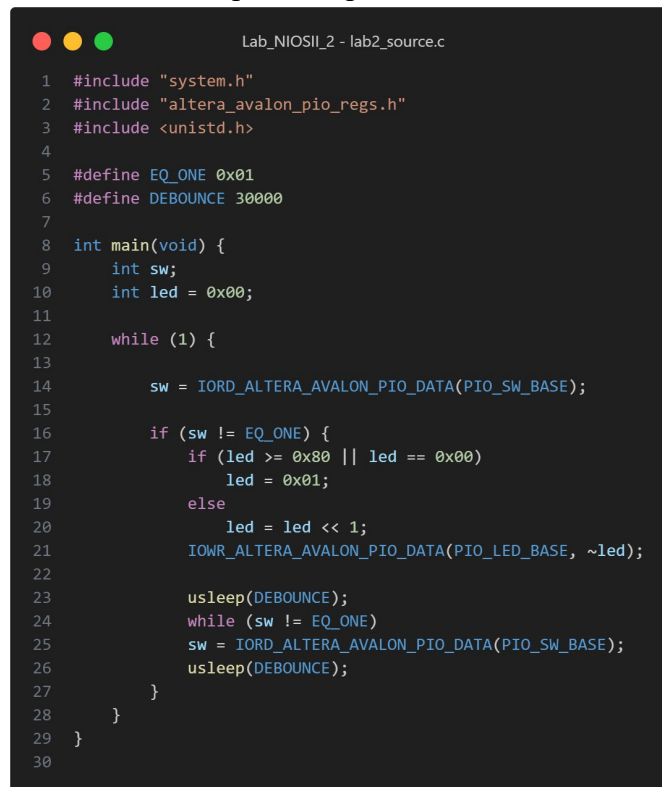


Рис. 23 – Синтаксис файла Lab2_source.c

Компиляция проекта в Eclipse

Выполним Lab1_sw → Build Project:

```
nios2-elf-g++ -T'../lab2_sw_bsp//linker.x' -msys-crt0='../lab2_sw_bsp//obj/HAL/src/crt0.o'
nios2-elf-insert lab2_sw.elf --thread_model hal --cpu_name nios2_PD --qsys true --simulation
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]
```

18:44:25 Build Finished (took 5s.293ms)

Рис. 24 – Build Project

При создании платформы, программой и данными инициализации занято 4616 Байт, свободно 10 КБ.

Выполним Lab1_sw_bsp → Nios II Lab1 → BSP Editor и в окне настроек зададим параметры следующим образом:

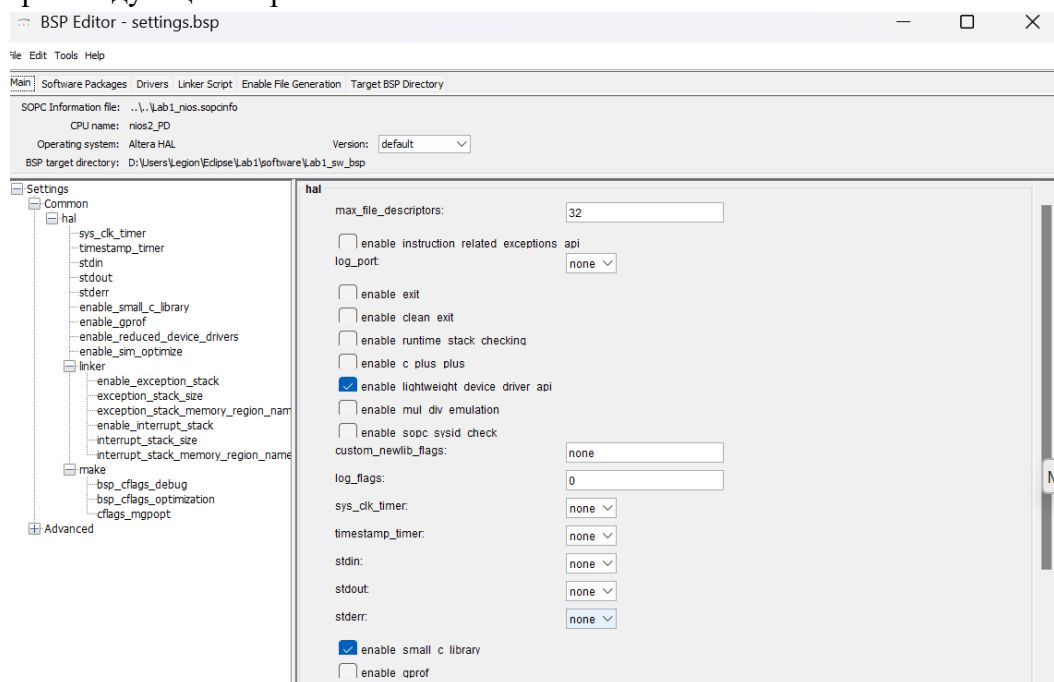


Рис. 25 – Настройка BSP Editor

Скомпилируем проект. Для этого выполним Lab1_sw → Build Project, проверим, что останется больше свободной памяти:

```
CDT Build Console [lab2_sw]
Info: Compiling lab2_source.c to obj/default/lab2_source.o
nios2-elf-gcc -xc -MP -MMD -c -I../lab2_sw_bsp//HAL/inc -I../lab2_sw_bsp/ -I../lab2_sw_bsp//drivers/inc -pipe -D__hal
Info: Linking lab2_sw.elf
nios2-elf-g++ -T'../lab2_sw_bsp//linker.x' -msys-crt0='../lab2_sw_bsp//obj/HAL/src/crt0.o' -msys-lib=hal_bsp -L../lab
nios2-elf-insert lab2_sw.elf --thread_model hal --cpu_name nios2_PD --qsys true --simulation_enabled false --stderr_de
Info: (lab2_sw.elf) 1632 Bytes program size (code + initialized data).
Info: 14 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]

19:12:34 Build Finished (took 5s.138ms)
```

Рис. 26 – Build Project после изменения настроек BSP Editor

Заметим, что памяти, оставшейся свободной (из ОЗУ 16 Кбайт, указанных при создании платформы, программой и данными инициализации занято 1632 Байт (было - 4616 Байт), свободно 14кБайт (было - 10кБайт).

Полная компиляция проекта в QP

Создадим Lab2.sdc файл с временными требованиями так, как показано на Рис. 40 ниже:

```
#####
# 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}]
```

Рис. 27 – SDC файл

Добавим файл к проекту:

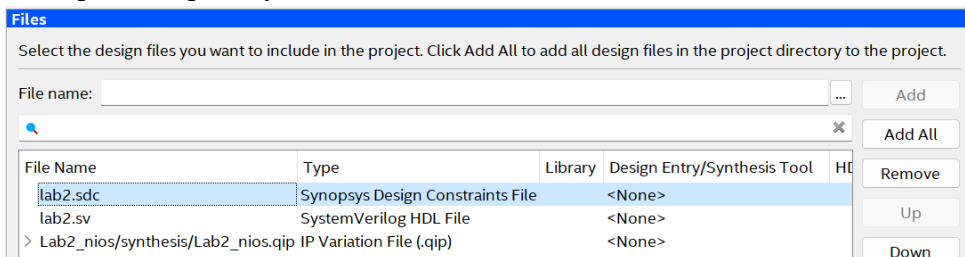


Рис. 28 – Добавление SDC файла

Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

Slow 1200mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	50.38 MHz	50.38 MHz	clock	
2	113.28 MHz	113.28 MHz	altera_reserved_tck	

Рис. 29 – Временные характеристики устройства

2.7. Реализация на плате

Конфигурирование FPGA

Подключим плату miniDilabCIV к ПК, включим питание и установим все переключатели sw[7:1] в состояние 0, а переключатель SW[0] в положение 1.

Запустим конфигурирование проекта на плате средствами Programmer.

Загрузка ПО

В приложении Eclipse выполним Run → Run as → NiosII Hardware:

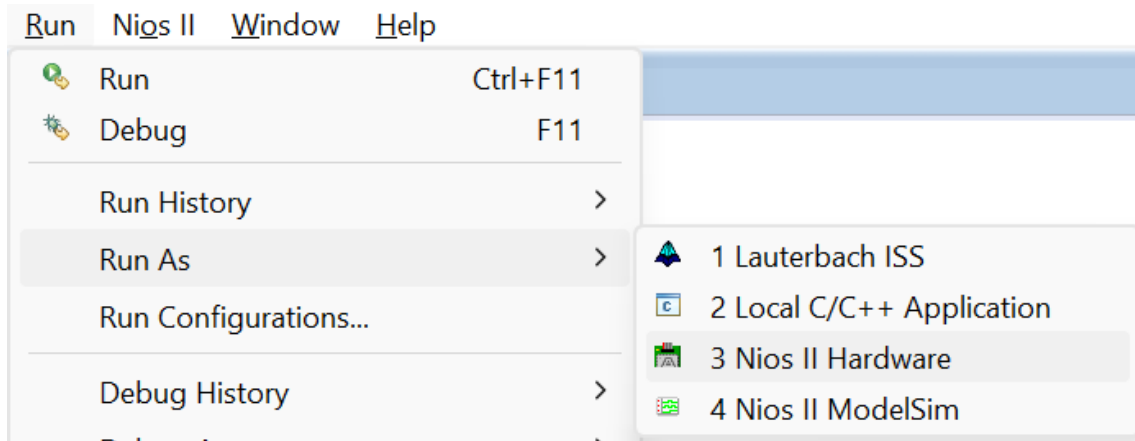


Рис. 30 – Загрузка ПО проекта средствами Eclipse

В появившемся окне выполним Target Connection → Refresh Connections (будет установлен имеющийся JTAG кабель и реализовано подключение к процессору NIOSII в FPGA):

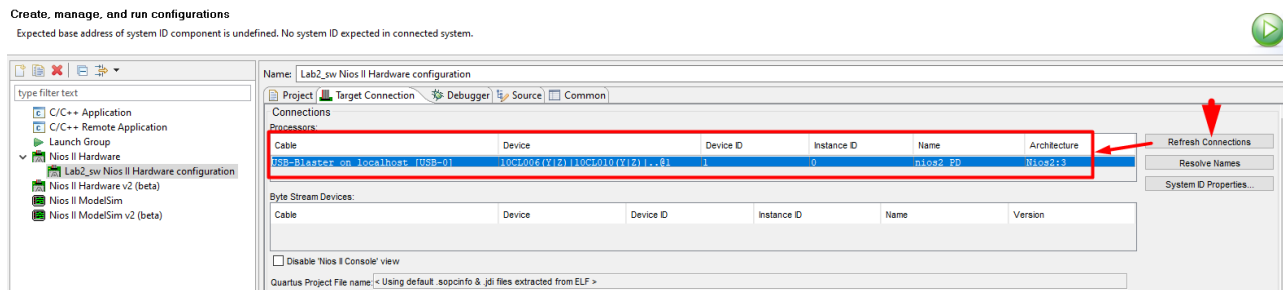


Рис. 31 – Refresh Connections

Запуск на плате

Повторно средствами Eclipse выполним команду Run → Run as → NiosII Hardware.

Проверим корректность работы проекта на плате.

Как и ожидалось, при переключении SW[0] из положения 1 в 0, загорится светодиод LED[0]. При нескольких переключениях SW[0] из 0 в 1 а затем в 0, будет включаться один из светодиодов в следующей последовательности: led[0]=>led[1]=>led[2]=>...=>led[7]=>led[0]=>... . Наконец, переключим SW[0] в положение 1.

Отладка ПО

В приложении Eclipse выполним Run → Debug as → NiosII Hardware.

После успешной загрузки приложения появится окно с сообщением - “переключиться GUI к виду ОТЛАДКА?”: нажмём кнопку Yes. Приложение будет запущено и пользовательский интерфейс открыт в режиме отладки NiosII Debug:

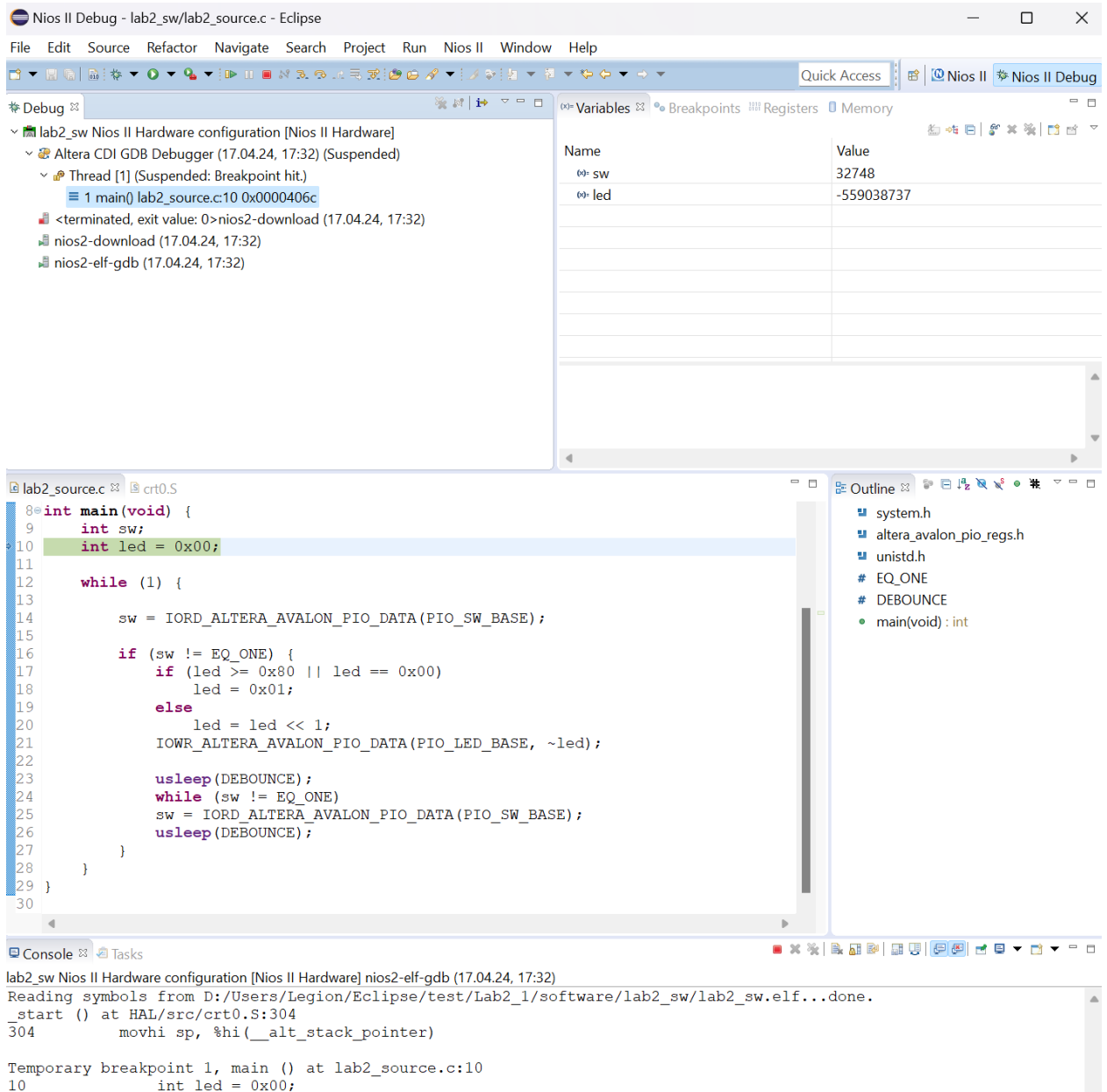


Рис. 32 – Режим отладки NiosII Debug

Повторный запуск на плате

Выполним Run → Resume и проверим корректность работы проекта на плате:

Переключим SW[0] из положения 1 в 0 - включится светодиод led[0]. Несколько раз переключим SW[0] из 0 в 1, а затем в 0. При этом будет включаться один из светодиодов в следующей последовательности: led[0]=>led[1]=>led[2]=>...=>led[7]=>led[0]=>.... . Наконец, переключим SW[0] в положение 1.

Выполним команду Run → Terminate, чтобы остановить программу. Теперь запустим режим отладки, нажав Run → Debug. Включим опцию отображения номеров в списке: Window → Preferences, General → Editors → Text Editor → вкл. Show line numbers.

Установим точки прерывания в строках 26 и 29 так, как показано на Рис. 33 ниже:

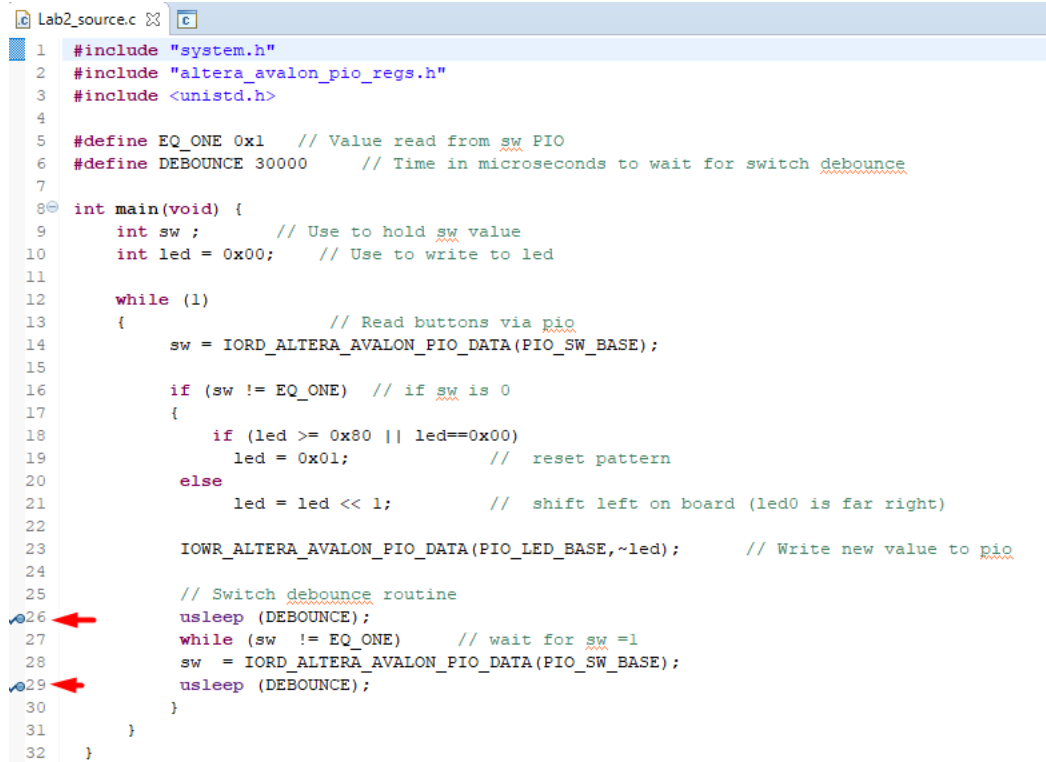


Рис. 33 – Установка точек прерывания

Выберем закладку Memory и в поле Monitors выполним ЛКМ → Add Memory Monitor. В окне Monitor Memory введём указатель &led:

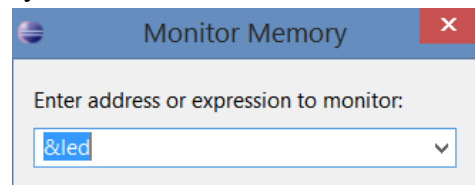


Рис. 34 – Установка указателя &led

Теперь в том же окне Monitor Memory введём указатель &sw. После этого запустим процессор, он будет находиться в ожидании переключения sw[0] в 0.

Переключим sw[0] в 0, на плате включится светодиод led[0]. Процессор остановится в первой точке прерывания (та, которая была указана нами ранее на строке 26). При этом обратим внимание на закладку Variables:

(x)= Variables		Breakpoints	Registers	Memory
Name	Value			
(x)= sw	0			
(x)= led	1			

Рис. 35 – Закладка Variables (1)

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

Теперь рассмотрим закладку `Memory`:

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

Рис. 36 – Закладка `Memory` (1)

По адресу переменной `sw` (адрес 4-7) содержится значение в формате `hex`, полученное с `sw[7:0]`. По адресу переменной `led` (адрес 8-B) содержится число в аналогичном формате `hex`, передаваемое на светодиоды `led7...led0`.

Теперь снова переключим `sw[0]` в положение 1, нажмём `resume` – при этом процессор запустится и остановится в следующей, указанной нами, точке прерывания (строка 29).

Теперь в закладке `Variables` мы сможем наблюдать за значением переменной `sw` (формат `int`), отображающей значение, полученное с `sw[0]`, а переменная `led` аналогичного формата будет отображать число, передаваемое на светодиоды `led8-led1`:

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

Рис. 37 – Закладка `Variables` (2)

В закладке `Memory` отображается информация о том, что по адресу переменной `sw` (адрес 4-7) содержит значение, полученное с `sw[7:0]`; по адресу переменной `led` (адрес 8-B) содержится число, передаваемое на светодиоды `led7...led0`:

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

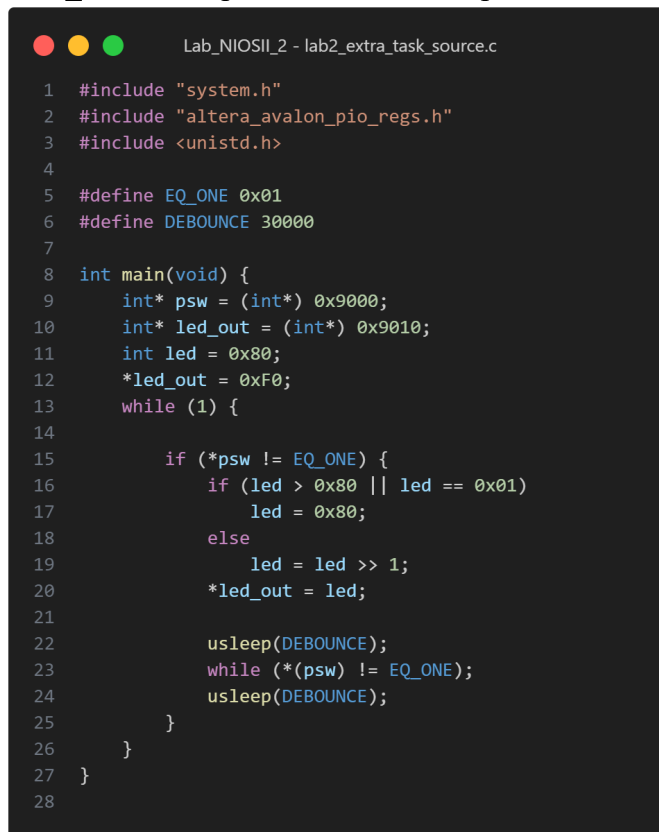
Рис. 38 – Закладка `Memory`

Данная процедура была проведена несколько раз, смещение светодиода происходило корректно, работа устройства была продемонстрирована преподавателю.

3. Дополнительное задание

3.1. Создание программного файла Lab2_extra_task2_source.c

Изменим текст программы так, чтобы использовать указатели (абсолютные адреса элементов PIO_SW и PIO_LED, которые были заданы при создании системы в пакете):



```
1 #include "system.h"
2 #include "altera_avalon_pio_regs.h"
3 #include <unistd.h>
4
5 #define EQ_ONE 0x01
6 #define DEBOUNCE 30000
7
8 int main(void) {
9     int* psw = (int*) 0x9000;
10    int* led_out = (int*) 0x9010;
11    int led = 0x80;
12    *led_out = 0xF0;
13    while (1) {
14
15        if (*psw != EQ_ONE) {
16            if (led > 0x80 || led == 0x01)
17                led = 0x80;
18            else
19                led = led >> 1;
20            *led_out = led;
21
22            usleep(DEBOUNCE);
23            while (*(psw) != EQ_ONE);
24            usleep(DEBOUNCE);
25        }
26    }
27 }
28
```

Рис. 39 – Изменённый исходный файл

Теперь переход осуществляется по абсолютным адресам. Также, была изменена та часть кода, которая отвечала за сдвиг светодиода. Теперь сдвиг происходит не влево, а вправо.

3.2. Реализация на плате

Как и ожидалось, визуально можем наблюдать корректную работу сдвигающего регистра. При переключении SW[0] из положения 1 в положение 0, происходит сдвиг вправо, при этом адреса в закладке Memory соответствуют ожиданиям. Работа с устройством на плате была продемонстрирована преподавателю.

4. Вывод

В ходе лабораторной работы был осуществлен процесс создания и настройки системы (Рис. 1) на базе процессора NIOS II с использованием пакета Quartus Prime и Eclipse IDE. За основу реализуемого проекта была взята система, созданная в ходе предыдущей лабораторной работы. Реализация проекта включала в себя создание аппаратной части с помощью Platform Designer (PD), создание программной части с помощью среды разработки Eclipse, где был создан файл для инициализации модуля памяти программ процессора, а также реализация проекта на плате и его пошаговая обработка средствами Eclipse.

В ходе тестирования проекта на плате светодиоды LED8-LED1 отображали последовательный сдвиг регистра влево, который можно было наблюдать на плате в виде светодиода, «идущего» справа налево. Сдвиг происходил при переводе SW[0] из положения 1 в положение 0. Это говорит о корректности работы созданного устройства.

В дополнительном задании были реализованы различные функциональности, такие как изменить направления сдвига регистра, а также использование указателей для адресации данных (абсолютная адресация данных). Корректная работа дополнительного задания была продемонстрирована преподавателю на плате.

Анализ результатов показал успешную работу созданной системы на плате, а также корректное отображение данных на светодиодах в зависимости от значений на переключателе SW[0].

Полученные навыки работы с NIOS II могут быть полезны при создании проектов в области встраиваемых систем, таких как системы управления, обработки сигналов, автоматизации и других. Среда Eclipse помогает легко заниматься процессом отладки и отслеживать изменяющиеся адреса в режиме реального времени.