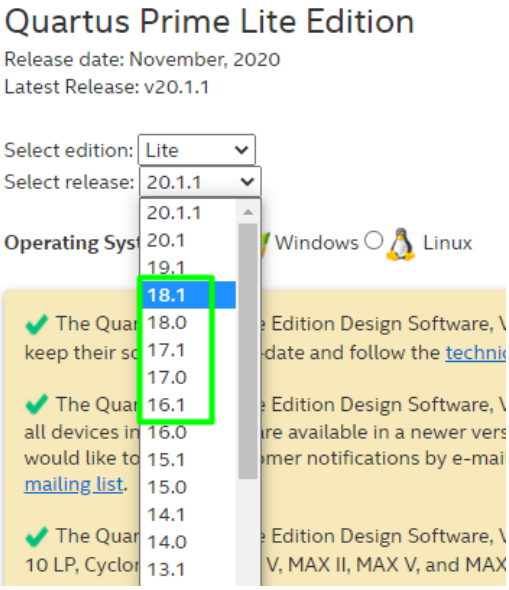
Задание labn_1s
(самостоятельное выполнение)

предполагается использование пакета QP Lite версии 16.1…18.1



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

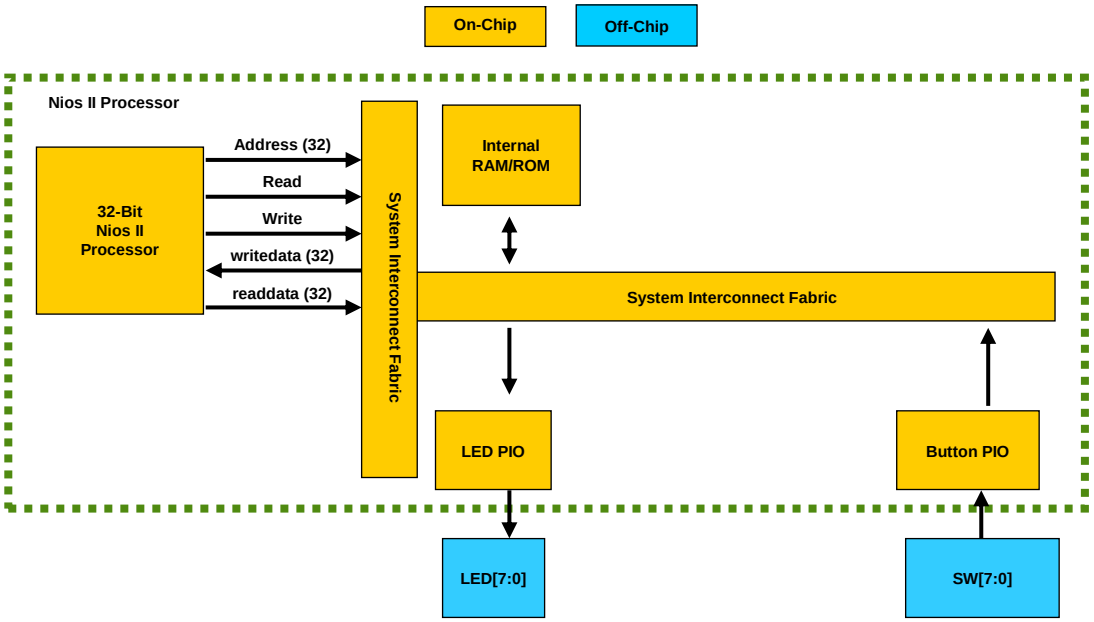
Введение

Цель упражнения – самостоятельно пройти процедуру 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 модуля счета, заданного переключателями SW[7..0].

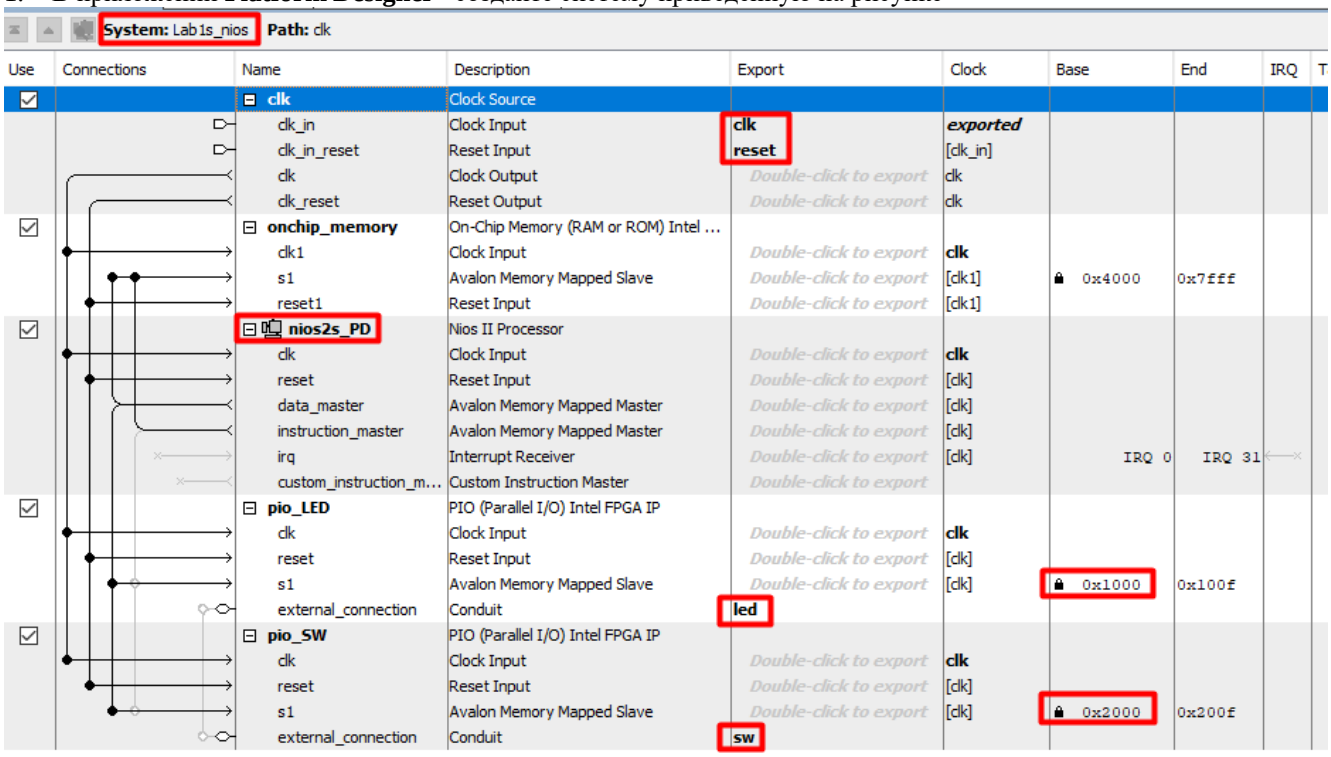


Часть 1 – Создание проекта

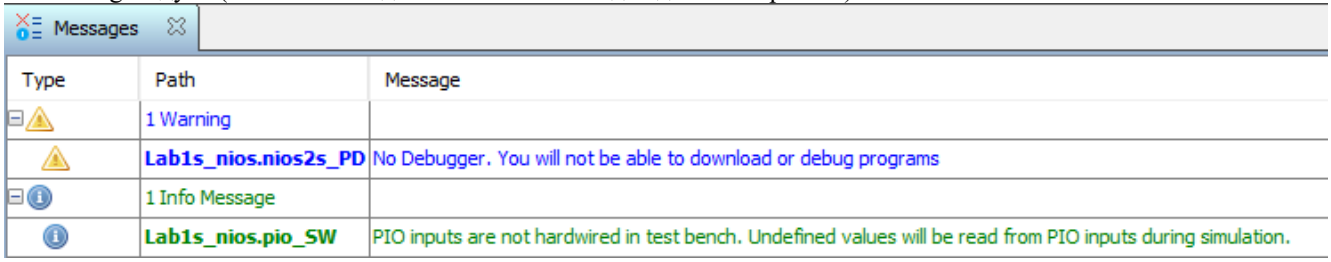
1. Создайте проект со следующими параметрами:
   - ✓ Рабочая папка *C:\Intel_trn\Q_NIOS\Lab1s*
   - ✓ Имя проекта: Lab1s
   - ✓ Имя модуля верхнего уровня в иерархии проекта: Lab1s
   - ✓ В окне **Add Files [page 2 of 5]** нажмите кнопку **Next.**
   - ✓ FPGA: **EP4CE6E22C8**.
   - ✓ Все остальные настройки – по умолчанию.

Часть 2 – Создание аппаратной части проекта

1. В приложении **Platform Designer** создайте систему приведенную на рисунке



2. Имя файла – Lab1s_nios.qsys
3. Модуль тактового сигнала: **clk**
   - ✓ Настройки такие же как в Lab1
4. Память: **onchip_memory**
   - ✓ Настройки такие же как в Lab1
5. Процессор: **nios2s_PD**
   - ✓ Настройки процессора такие же как в Lab1
6. Модуль вывода данных на светодиоды: **pio_LED**
   - ✓ Настройки такие же как в Lab1, позволяет выводить данные на 8 светодиодов
7. Модуль ввода данных с переключателей: **pio_SW**
   - ✓ Позволяет вводить данные с 8 переключателей sw[7..0]
8. Закладка Messages должна содержать одно предупреждение, информирующее о том, что не подключен JTAG Debug модуль (но это было сделано сознательно для данного проекта)



9. Запустите процедуру создания HDL описания системы.
10. Оставьте все значения по умолчанию и нажмите кнопку Generate.
11. Создание HDL описания системы должно завершится только с предупреждениями, касающимися отсутствия JTAG соединения для отладки.
12. В пакете QP подключите к проекту файл Lab1s_nios.qip

13. Создайте в текстовом редакторе файл (имя файла Lab1s.sv) верхнего уровня в иерархии проекта (для этого целесообразно использовать файл Lab1s_nios_inst.v из папки C:\Intel_trn\Q_NIOS\Lab1s\Lab1s_nios)

```
module Lab1s (
    input bit clk,          // Clock
    input bit [7:0] sw,     // data in
    input bit pbb,          // Asynchronous reset active low
    output bit [7:0] led
);

    Lab1s_nios u0 (
        .clk_clk         (clk), //   clk.clk
        .reset_reset_n   (pbb), //   reset.reset_n
        .led_export      (led), //   led.export
        .sw_export       (sw)   //   pbb.export
    );

endmodule
```

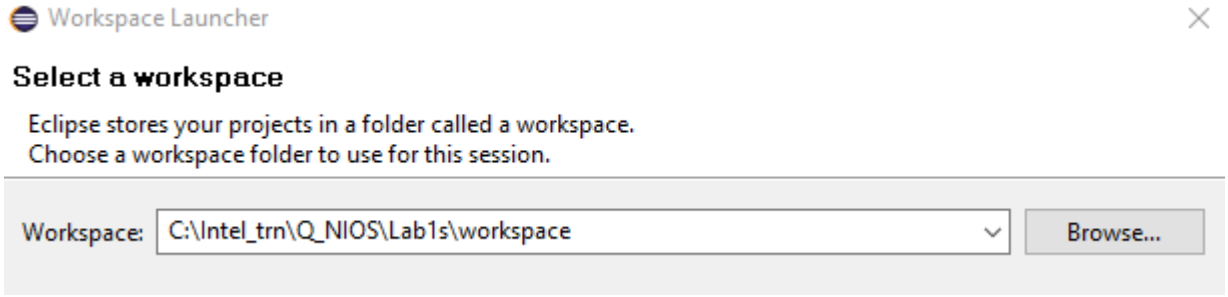
14. *Файл верхнего уровня может быть создан и в графическом редакторе*

15. Выполните команду Assignments=>Settings => Compilation Process Settings
    - ✓ Установите опцию Use all available processors
16. Проверка синтаксиса проекта.
    - ✓ Выполните команду Processing=>Start=>Start Analysis and Elaboration
    - ✓ Компиляция должна завершиться без ошибок
17. Назначение выводов проекта.
    - ✓ Запустите редактор назначения выводов (Pin Planner): Assignment=>Pin Planner.
    - ✓ Назначьте выводы так, как показано на рисунке ниже

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

Часть 3 – Создание программной части проекта

1. Запустите оболочку для разработки/отладки программ - NIOSII IDE/
   - ✓ Назначьте рабочую область для данной лабораторной



   - ✓ Нажмите кнопку OK

18. Выполните команду File=>New=>NIOS II Application and BSP from Tempate.
    - ✓ В окне помощника введите:
      1. В разделе Target Hardware Information: файл lab1s_nios.sopcinfo – файл с описанием созданной системы.
      2. В разделе Aplication Project: название проекта – Lab1s_sw
      3. В разделе Select Project Template: выберите Blank Project
19. Выполните команду File=>New=>Other.
    - ✓ Выберите Source File в категории C/C++.
    - ✓ В появившемся окне New source file:
      1. Найдите и укажите папку Lab1s_sw,
      2. введите название файла: Lab1s_source.c;
    - ✓ Выберите Template => Default C source template.
20. Введите текст программы на языке Си, обеспечивающий счет на сложение по модулю, заданному переключателями SW[7:0].
21. Пример кода приведен на рисунке

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 }
```

22. Выберите папку Lab1s_sw, нажмите правую клавишу мыши и укажите команду Build Project.
23. Зафиксируйте размер занятой и свободной областей памяти.
24. Уменьшение размера программы (так, как было сделано в лаборатории Lab1)
25. Выберите папку Lab1s_sw, нажмите правую клавишу мыши и укажите команду Build Project. Будет запущен компилятор.
26. Зафиксируйте размер занятой и свободной областей памяти. Сравните с предыдущими результатами.
27. Создание hex файла для инициализации памяти FPGA:
    - ✓ Выберите папку Lab1s_sw, нажмите правую клавишу мыши и укажите команду Make Targets => Build.
    - ✓ В появившемся окне выберите mem_init_generate и нажмите Build.

Часть 4 – Полная компиляция проекта

1. В пакете QP подключите к проекту файл с описанием файла инициализации памяти, созданного в NiosII IDE:
   - ✓ C:\Intel_trn\Q_NIOS\Lab1s\software\Lab1s_sw\mem_init\ meminit.qip

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

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[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 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 Lab1s.sdc к проекту.
   a. Выполните команду Assignment=>Settings
   b. В разделе Timing Analyzer добавьте файл Lab1s.sdc к проекту.
4. С помощью команды **Processing => Start Compilation** осуществите полную компиляцию проекта.

Часть 5 – Конфигурирование FPGA и проверка работы проекта на плате

1. Подключите плату miniDilabCIV к ПК.
2. Осуществите конфигурирование FPGA
3. Проверьте работу проекта на плате:

   a. Процессор NIOSII на светодиодах LED8 … LED1 отображает двоичные коды чисел от 0 до модуля счета, заданного переключателями SW[7..0].

Упражнение Lab1s завершено.