

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Базовая кафедра «Вычислительные технологии»

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Проектирование реконфигурируемых систем
на кристалле»
Тема: Прерывания в программно-аппаратных системах на основе
кристалла Cyclone V
Вариант 4

Студенты гр. 6309

Васин А. М.

Жвакин К. Э.

Ладыженский Р. С.

Преподаватель

Шарагина Н.С.

Санкт-Петербург

2021

Цель работы.

Цель работы состоит в изучении процесса проектирования систем с использованием аппаратных прерываний soft-процессора Nios II.

Основные теоретические положения.

Процессор Nios II поддерживает работу с прерываниями двумя способами. Первый способ основан на использовании внутреннего контроллера прерываний Nios II. Этот контроллер поддерживает 32 аппаратных прерывания, ядро процессора имеет 32 входа запроса прерываний (IRQ0 – IRQ31), для каждого источника прерываний используется уникальный вход. Приоритет прерываниям назначается программно, поддерживаются вложенные прерывания.

Настройка прерываний осуществляется установкой определенных битов в управляющих регистрах. Регистры, требующие настройки при организации прерываний, приведены в табл. 1.

Схема формирования запроса на прерывание soft-процессора на основании поступивших запросов и состояний регистров приведена на рис. 1.

Таблица 1

Регистры управления прерыванием

Регистр	Имя	b31 ...b2	b1	b0
ctl 0	<i>status</i>	резерв	U	PIE
ctl 1	<i>estatus</i>	резерв	EU	EPIE
ctl 3	<i>ienable</i>	Биты разрешения прерывания Биты возникших прерываний		
ctl 4	<i>ipending</i>			

Глобальное разрешение-запрет внешних прерываний осуществляется с помощью бита PIE в регистре *status*. Регистр *ctl1* хранит копию регистра состояния во время обработки прерываний. Запрет-разрешение отдельных прерываний процессорной системы осуществляется программно через контрольный регистр *ienable*, который содержит бит разрешения прерываний для каждого входа IRQ. Регистр содержит информацию о прерываниях, требующих обработки. Обработчик прерываний определяет, какие из них

имеют наибольший приоритет, и передает управление соответствующей подпрограмме.

Для генерации прерываний необходимо выполнение следующих условий:

- бит PИE регистра status установлен в единичное состояние;
- один из входов запросов прерывания (IRQ_k) процессора активирован;
- соответствующий бит регистра разрешения прерываний ienable содержит единичное значение.

Второй способ реализации прерываний предполагает использование внешнего векторного контроллера прерываний (VIC, Vectored Interrupt Controller), например представленного в библиотеке периферийных модулей среды создания реконфигурируемых систем на кристалле SOPC Bulder.

Векторный контроллер позволяет существенно сократить время отклика процессора на прерывания, что актуально для систем реального времени. Один внешний контроллер позволяет обслуживать до 32 запросов прерываний, причем имеется возможность их каскадирования, что позволяет увеличить количество запросов. Контроллер подключается к процессору через интерфейс внешнего контроллера прерываний (EIC, External Interrupt Controller Interface). Если в систему включен внешний векторный контроллер прерываний, внутренний контроллер прерываний не реализуется, а SOPC Builder подключает прерывания к внешнему.

При поступлении нескольких запросов на прерывания внешний векторный контроллер на основании принятой системы приоритетов выбирает то, которое будет обрабатываться и передает ядру Nios II адрес соответствующего обработчика. Некоторые внешние прерывания могут быть сконфигурированы как немаскируемые (NMI, Nonmaskable Interrupt) – они не блокируются битом PИE регистра status и не имеют уровня.

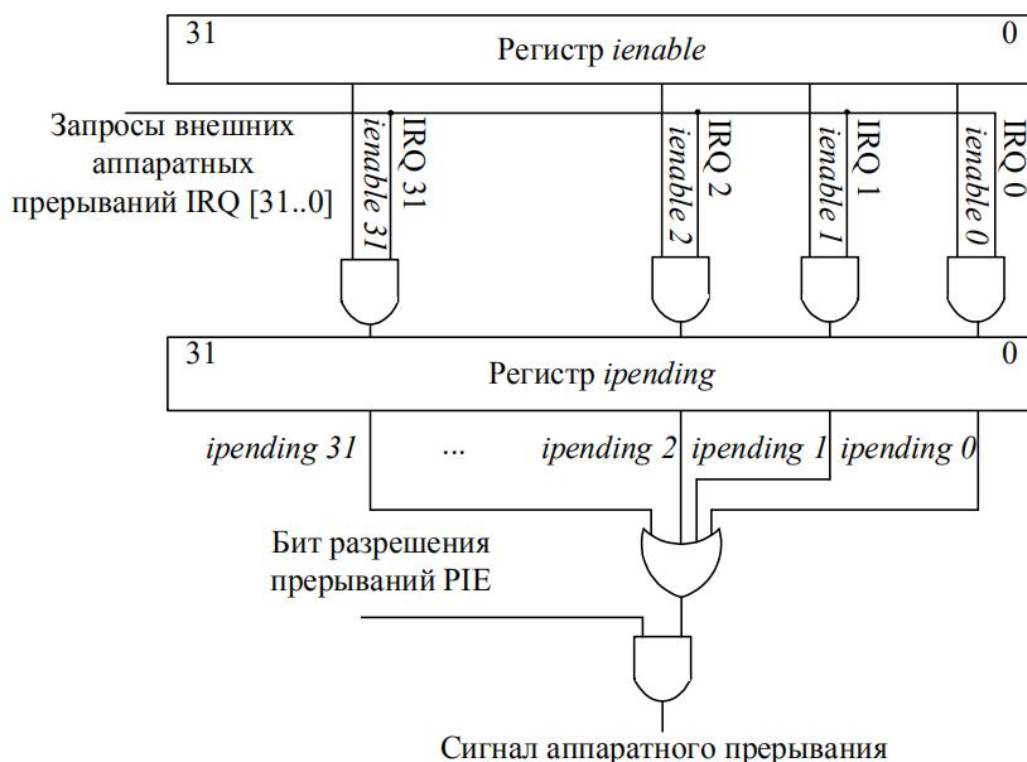


Рис. 1 - Схема генерации прерываний soft-процессора Nios II

При работе с прерываниями программное обеспечение состоит из двух частей: основной программы и функций, обеспечивающих работу в режиме прерываний. Функции должны реализовывать инициализацию прерывания, запреты прерываний, а также основные действия по обработке запросов IRQ.

При обработке прерывания необходимо учитывать асинхронность запроса. Ресурсы, используемые основной программой, не должны искажаться прерывающей функцией. Кроме создания собственно программ проектировщик должен задать места расположения фрагментов. При создании системы часто необходима определенная настройка приоритетов системы прерываний.

Задание на работу.

Создать и отладить аппаратно-программную систему, в которой основная программа перемещает светящийся элемент по светодиодной линейке и изменяет направление перемещения при срабатывании прерывания. Источником прерывания служит передний и задний фронт входного сигнала (тактовая кнопка или движковый переключатель).

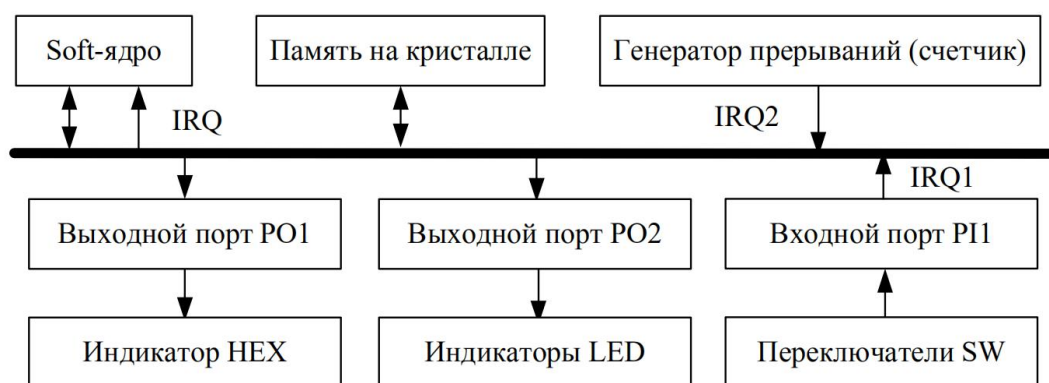


Рис. 2 - Схема проектируемого устройства

Выполнение работы.

Этап 1. Разработка системы

1. Выполним конфигурирование процессорной части СнК, а также сгенерируем файлы описания soft-процессора NIOS II со встроенной в кристалл памятью программ и данных и блоком индикации. Выполним компиляцию проекта, назначим контакты, выполним загрузку процессорного ядра в ПЛИС аналогично этапам, описанным в прошлой лабораторной работе.

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		clk_0	Clock Source	clk	clk_0			
		clk_in	Clock Input	reset				
		clk_in_reset	Reset Input					
		clk	Clock Output	Double-click to export				
		clk_reset	Reset Output	Double-click to export				
<input checked="" type="checkbox"/>		nios2_qsys_0	Nios II Processor					
		clk	Clock Input	Double-click to export	clk_0			
		reset_n	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]			
		jtag_debug_module_re...	Reset Output	Double-click to export	[clk]			
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]			
		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]	0x0001_0800	0x0001_0fff	
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	Double-click to export	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000_8000	0x0000_celf	
		reset1	Reset Input	Double-click to export	[clk1]			
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1020	0x0001_1027	
<input checked="" type="checkbox"/>		pio_0	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1000	0x0001_100f	
		external_connection	Conduit	Double-click to export				
<input checked="" type="checkbox"/>		pio_1	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1010	0x0001_101f	
		external_connection	Conduit	Double-click to export				
				interrupt_btn				

Рис. 3 - Результат конфигурирования системы

2. Произведём настройку порта ввода, включив генерацию прерывания по любому фронту (галочка Generate IRQ, выбор IRQ type и выбор edge type) в

соответствии с заданием.

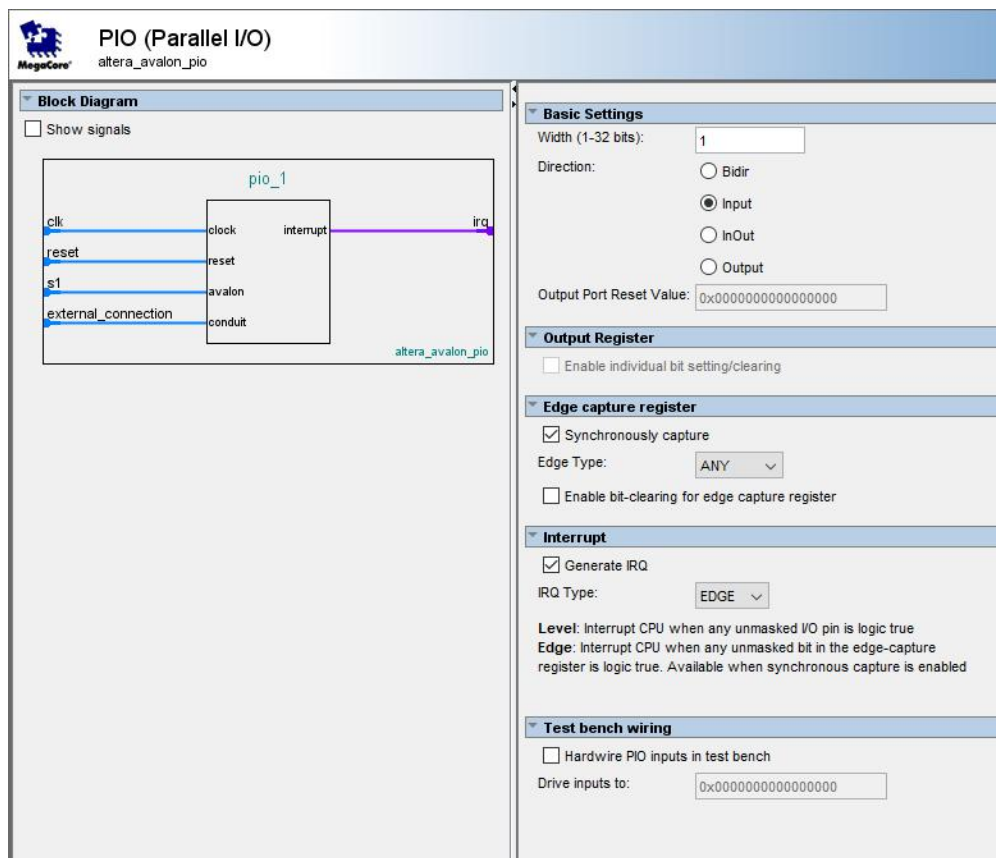


Рис. 4 - Настройки портов PIO

3. Установим прерывания от линии порта, в соответствии с руководством, приведённом в методическом указании к лабораторной работе. Дополнительно установим приоритеты прерываний.

4. По завершению настроек, выполним генерацию системы, а также конфигурирование системы на кристалле ПЛИС, в соответствии с последовательностью действий приведённых в предыдущих лабораторных работах.

Этап 2. Разработка программного обеспечения

1. Основная программа осуществляет бегущий огонь вкруговую и по прерыванию бегущий огонь изменяет своё направление. Листинг программы приведён ниже.

Листинг 1. Исходный код программы

```
#include "sys/alt_stdio.h"
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include <unistd.h>
```

```

#include "sys/alt_irq.h"
volatile int edge_capture;

int flag_irq;

void init_pio();

int main()
{
    alt_putstr("Hello from Nios II!\n");

    init_pio();
    edge_capture = 0;
    int data_led = 1;
    int direction = 1;
    //unsigned long long i = 0;
    /* Event loop never exits. */
    while (1)
    {
        if(flag_irq == 1)
        {
            flag_irq = 0;
            direction = (direction) ? 0 : 1;
            data_led = 1;
        }

        data_led = (!direction) ? data_led << 1 : data_led >> 1;
        data_led = (data_led > 8 || data_led == 0) ? ((direction) ? 8 : 1) : data_led;

        IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE, ~data_led);

        usleep(100000);
    }

    return 0;
}

#ifdef ALT_ENHANCED_INTERRUPT_API_PRESENT
static void handle_btn_irq (void* context)
#else
static void handle_btn_irq (void* context, alt_u32 id)
#endif
{
    volatile int* edge_capture_ptr = (volatile int*) context;
    *edge_capture_ptr += 1;
    flag_irq = 1;
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PIO_1_BASE, 0); //reset irq flag
}

void init_pio()
{
    void* edge_capture_ptr = (void*) &edge_capture;

    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PIO_1_BASE, 0x1); //enable irq
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PIO_1_BASE, 0); //reset irq flag

#ifdef ALT_ENHANCED_INTERRUPT_API_PRESENT
    alt_ic_isr_register (PIO_1_IRQ_INTERRUPT_CONTROLLER_ID, PIO_1_IRQ, handle_btn_irq,
        edge_capture_ptr, 0x0);
#else
    alt_irq_register (PIO_1_IRQ, edge_capture_ptr, handle_btn_interruptions);
#endif
}

```

```
#endif  
}
```

В данной работе было реализовано три функции: главная, инициализация порта ввода-вывода и функция call-back, которая срабатывает по прерыванию и выставляет флаг прерывания для дальнейшей реакции в главном цикле программы.

Этап 3. Проверка работы системы на отладочной плате

После распиновки была произведена имплементация СнК в ПЛИС отладочной платы Terasic SocKit. Также работоспособность спроектированного устройства была проверена.

Ниже на рисунке представлены затраты на реализацию проекта.

Flow Status	Successful - Sun Oct 24 12:18:06 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	lab_4
Top-level Entity Name	lab_4
Family	Cyclone V
Device	5CSXFC6D6F31C8ES
Timing Models	Preliminary
Logic utilization (in ALMs)	808 / 41,910 (2 %)
Total registers	1025
Total pins	7 / 499 (1 %)
Total virtual pins	0
Total block memory bits	171,264 / 5,662,720 (3 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI PMA RX ATT Deserializers	0
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total HSSI PMA TX ATT Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

Рис. 5 - Оценка затрат на реализацию проекта

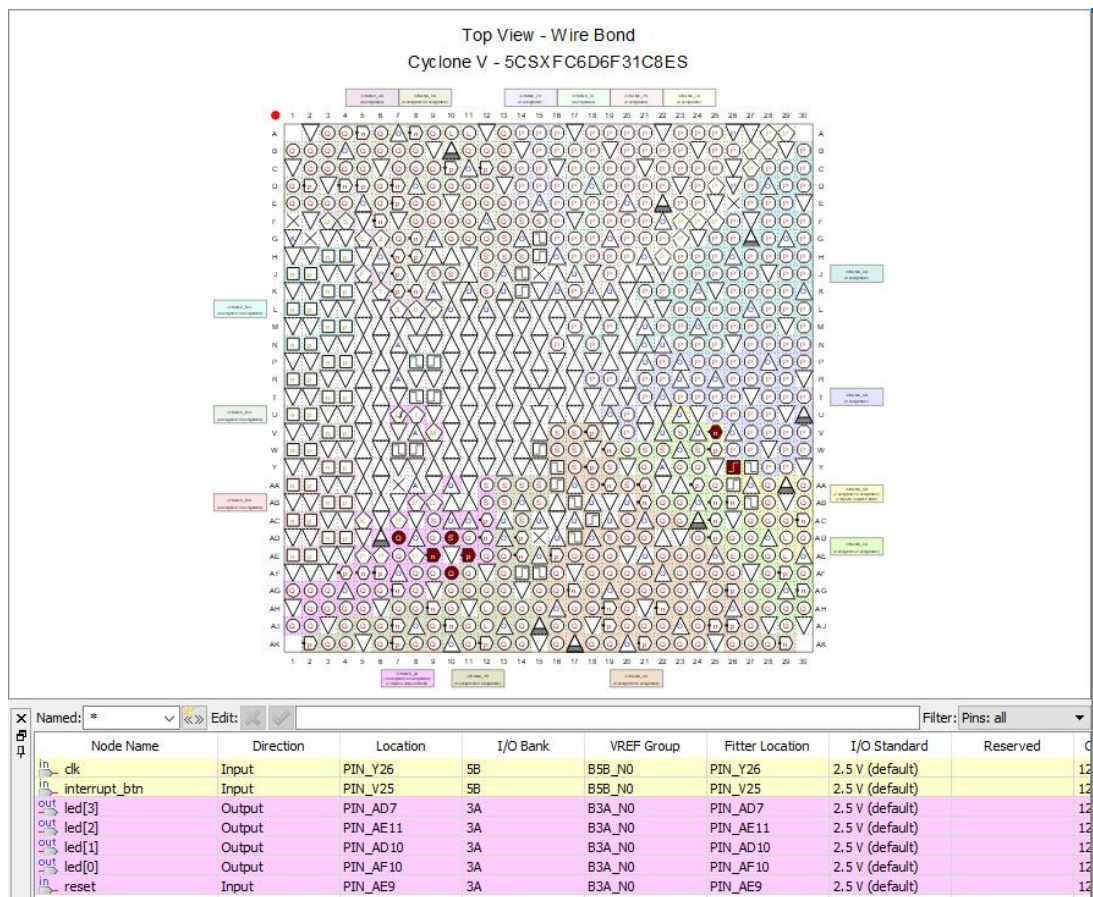


Рис. 6 - Назначение входов и выходов проекта на контакты ПЛИС

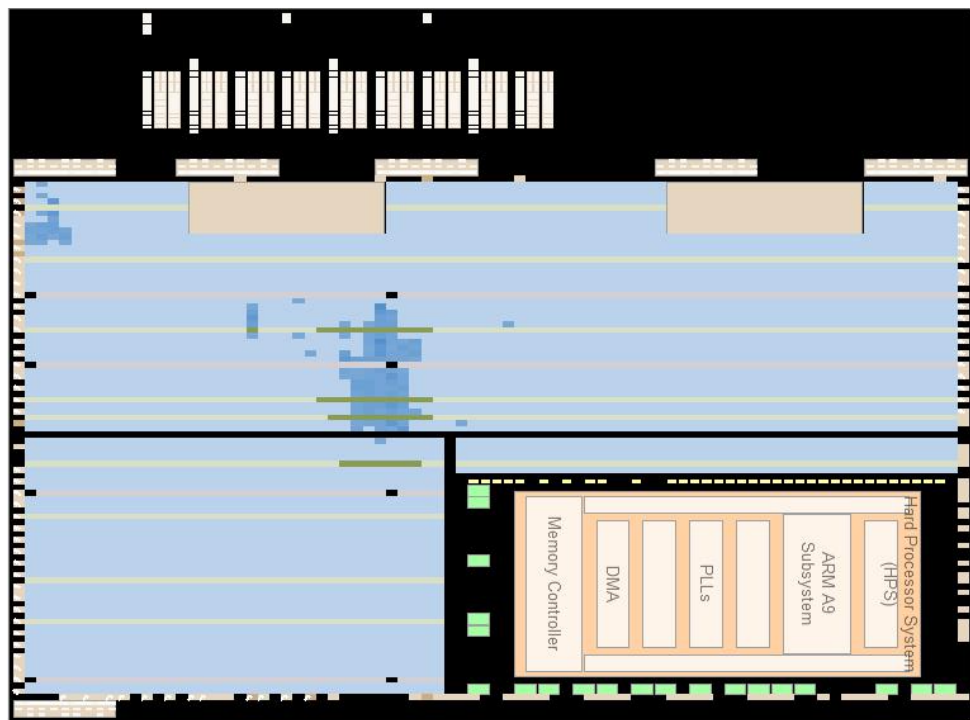


Рис. 7 - Расположение проекта в заданной ПЛИС

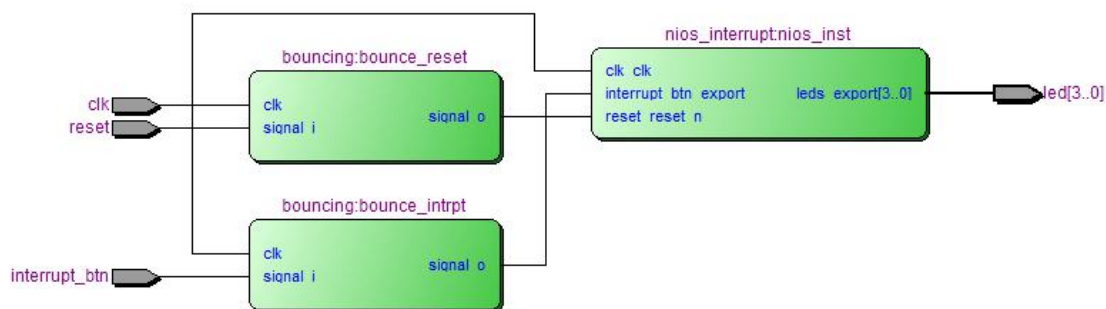


Рис. 8 - RTL-представление проекта

Выводы.

В ходе выполнения данной лабораторной работы был изучен процесс проектирования систем с использованием аппаратных прерываний soft-процессора Nios II, а также спроектирована аппаратно-программная система, которая перемещает светящийся элемент по светодиодной линейке и по прерыванию (по обоим фронтам) изменяет направление перемещения на противоположное.