

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

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Проектирование реконфигурируемых систем**  
**на кристалле»**  
**Тема: Имплементация проектов в реальные ИС**  
**Вариант 5**

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 получении навыков создания проекта, его моделирования, имплементации в ПЛИС с последующей внутрикристальной отладкой проекта с использованием встроенного в САПР логического анализатора.

### **Основные теоретические положения.**

Один из способов оценки корректности проектов основан на моделировании, предполагающем отладку проекта на модели путем анализа реакций разрабатываемых схем на стимулирующие воздействия. Несмотря на высокую вероятность обнаружения имеющихся дефектов моделирование не всегда позволяет оценить работу схемы. Более эффективны методы, основанные на экспериментах с реальным оборудованием. Получившие в последнее время широкое распространение прототипные платы разработчика, содержащие ПЛИС, позволяют организовать подобные эксперименты.

Для получения навыков работы с реальной ПЛИС предлагается провести модельную отладку проекта, представленного исходным описанием на языке VHDL, имплементацию проекта в ПЛИС учебного стенда и анализ работоспособности полученной схемы.

Отладка созданного проекта будет состоять из двух этапов: моделирования проекта и отладки на реальной схеме.

Отладка проекта на реальной схеме будет производиться с помощью встроенного в пакет Quartus II логического анализатора Signal Tap II Logic Analyzer. Использование встроенного логического анализатора – это один из способов внутрикристальной отладки проекта, возможный при наличии неиспользованных ресурсов ПЛИС. Созданные в процессе проектирования логические анализаторы загружаются в ПЛИС и подсоединяются к интересующим разработчика цепям; таким образом, разработчик имеет возможность наблюдать за реальными состояниями различных сигналов, фиксируемых логическим анализатором. Signal Tap II позволяет создавать и встраивать в проект определенное число

логических анализаторов, оперативно изменять условия фиксации данных в их памяти и отображать эти данные на экране компьютера.

### **Задание на работу.**

Разработать на языке System Verilog описание комбинационной схемы, выполняющей функции соответствующие выбранному варианту задания. Произвести моделирование проекта, выполнить его имплементацию в ПЛИС и отладку на реальной схеме.

Реализовать 10-разрядный регистр сдвига влево, реагирующий на каждое нажатие клавиши «сдвиг», с выводом выходной информации на семисегментные индикаторы. Значение вдвигаемого старшего разряда задается переключателем «данные».

### **Выполнение работы.**

#### ***Этап 1. Создание проекта в САПР Quartus II***

Был создан новый проект и заданы семейство и тип микросхемы: 5CSXFC6D6F31C8. Подготовлено описание схемы в соответствии с вариантом задания, для чего был создан новый SV-файл.

#### **Листинг 1. Исходный код описания схемы.**

```
module shift_reg(
    input logic btn_i,           //Trigger to shift
    input logic digit_i,        //Data shifted
    input logic clk,            //Input clk
    input logic reset,          //Input reset
    output logic [3:0] leds,
    input logic direction       //Direction of shifting
);
assign leds = data_reg[3:0];

logic [9:0] data_reg;
logic prev_btn;

logic [20:0] clock_counter;

logic btn;
logic digit;

assign btn = ~btn_i;
assign digit = ~digit_i;
```

```

assign clk_o = clk;

logic nreset;
assign nreset = ~reset;

logic clock_18;

assign clock_18 = clock_counter[18];

always @(posedge clk)
begin
    clock_counter++;
end

always @(posedge clk, posedge nreset)
begin
    if(nreset)
    begin
        data_reg = 10'd15;
    end
    else
    begin
        if(btn)
        begin
            if(!prev_btn)
            begin
                prev_btn = 1'b1;
                data_reg = direction ? {6'd0, data_reg[2:0], digit} :
{6'd0, digit, data_reg[9:1]};
            end
        end
        else
        begin
            prev_btn = 1'b0;
        end
    end
end
end

endmodule

```

Далее было выполнено назначение для каждого входного и выходного сигнала проекта контактов ПЛИС, результат представлен на рисунке 1.

После назначения контактов была выполнена повторная компиляция проекта.

Оценка затрат на реализацию проекта представлена на рисунке 2.

На рисунке 3 представлен RTL-вид проекта.

Расположение проекта в заданной ПЛИС представлено на рисунке 4.

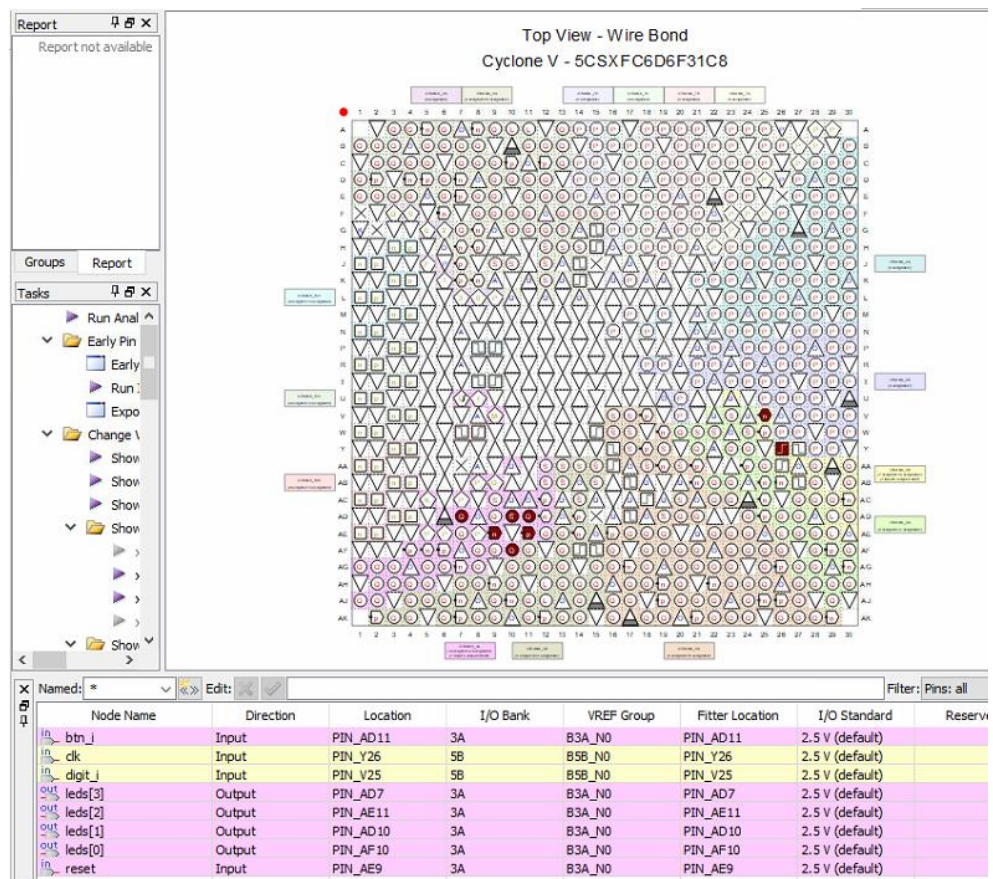


Рисунок 1 – Назначение входов и выходов проекта на контакты ПЛИС

Flow Status	Successful - Wed Oct 13 18:43:49 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	shift_reg
Top-level Entity Name	shift_reg
Family	Cyclone V
Device	5CSXFC6D6F31C8
Timing Models	Preliminary
Logic utilization (in ALMs)	87 / 41,910 ( < 1 % )
Total registers	73
Total pins	8 / 499 ( 2 % )
Total virtual pins	0
Total block memory bits	0 / 5,662,720 ( 0 % )
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 % )

Рисунок 2 – Оценка затрат на реализацию проекта

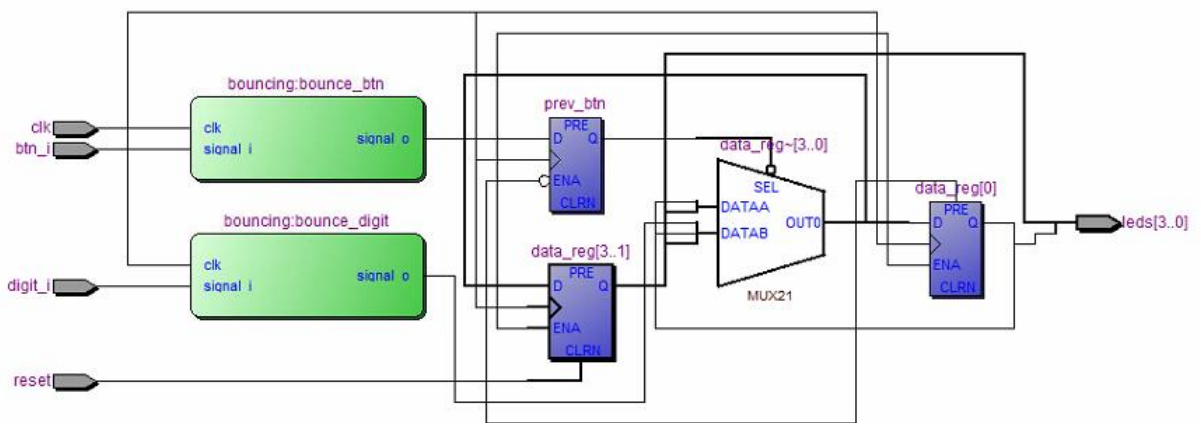


Рисунок 3 – RTL-представление проекта

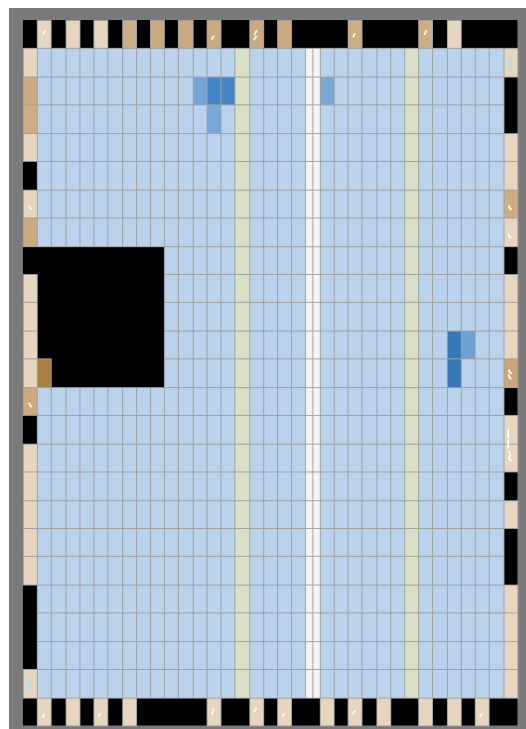


Рисунок 4 – Расположение проекта в заданной ПЛИС

## ***Этап 2. Моделирование проекта***

Для моделирования проекта был создан файл `tb_shift.reg`, в котором были описаны входные воздействия. Результаты моделирования представлены на рисунке 5.

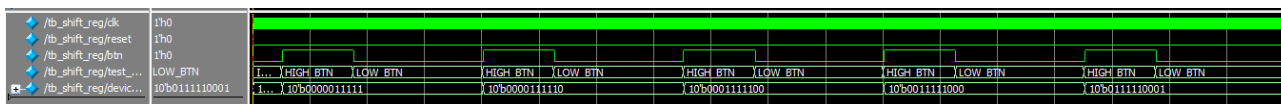


Рисунок 5 – Результат моделирования проекта

На рисунке 7 видно, что значение данных изменяется согласно заданию на переднем фронте сигнала clk при высоком уровне сигнала in.

### ***Этап 3. Загрузка проекта в ПЛИС***

Учебная плата была подключена к компьютеру через программатор и проект был загружен в ПЛИС.

### ***Этап 4. Отладка проекта на реальной схеме***

В результате нажатия на кнопку на плате, значение данных менялось не тем образом как при моделировании. Это происходило из-задребезга контактов – каждый скачок происходит изменение значения данных, а таких скачков при нажатии кнопки непредсказуемое количество. Причемдребезг происходит как при нажатии, так и при отпуске кнопки.

### ***Этап 5. Определения интервала дребезга контактов***

Для определения интервала дребезга был запущен логический анализатор SignalTap II. Затем был выбран сигнал тактирования анализатора clk (10 МГц), создан список сигналов, необходимых для отладки схемы, а также выбрано количество записываемых сэмплов логическим анализатором (64к):

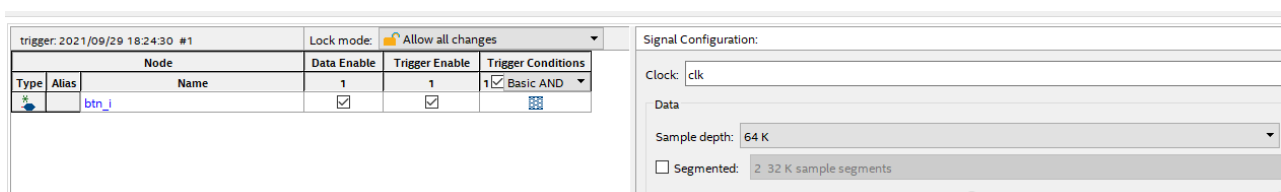


Рисунок 6 – Задание списка сигналов и определение сигнала тактирования в Signal Tap II

Проект снова был скомпилирован и загружен в ПЛИС с помощью Signal-Tap II Logic Analyser. Было задано условие срабатывания логического анализатора в правой области окна настройки:

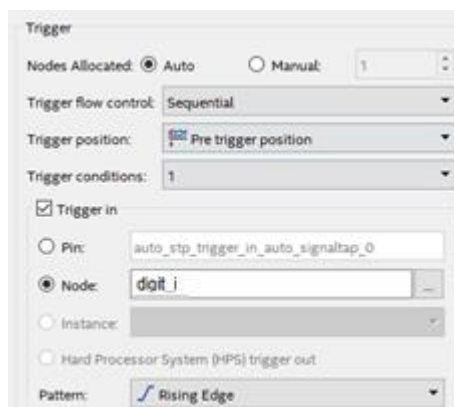


Рисунок 7 – Задание условия срабатывания логического анализатора

Логический анализатор был запущен в работу, и была получена следующая временная диаграмма:

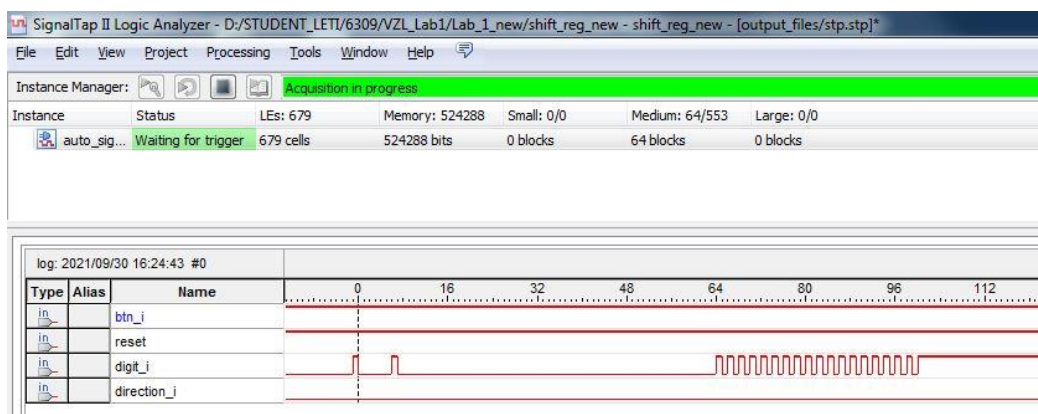


Рисунок 8 – Временная диаграмма зафиксированных сигналов


В результате работы логического анализатора мы можем наблюдать дребезг контактов на входном сигнале, который подключен к выводу с тактовой кнопки.

Было выполнено 4 измерения времени дребезга.

Таблица 1. Измерение времени дребезга контактов.

N	Временная диаграмма	Время дребезга, тактов clk
1		1281
2		446



3		840
---	--	-----

Максимальное время дребезга составляет 1300 тактов (округляя) сигнала clk. Исходя из того, что это 1300 тактов генератора 10МГц, можно сказать, что максимальное время дребезга составляет  $1300 * 5 = 6500$  тактов основного генератора (50МГц).

### ***Этап 6. Устранение дребезга контактов***

Для устранения влияния дребезга в основной проект был создан дополнительный модуль, в котором происходит устранение дребезга.

Листинг 2. Исходный код схемы устранения дребезга.

```
`define LEVEL_COUNT 32'd10000
module bouncing (
    input clk,
    input signal_i,
    output logic signal_o
);
    logic prev_signal;
    logic signal;
    assign signal = signal_i;
    int counter;

    always @(posedge clk)
    begin
        if (prev_signal == signal)
            begin
                counter++;
            end
        else
            begin
                prev_signal = signal;
                counter = 0;
            end
        if(counter == `LEVEL_COUNT)
            begin
                counter = 0;
                signal_o = signal;
            end
    end
endmodule
```

В исходном коде листинга 2 видно, что при изменении входного сигнала происходит запуск счетчика и если счетчик досчитал до числа, определенного дефайном, то это значит, что входной сигнал устоялся и выходной становится таким же как входной.

Тестбенч для моделирования был доработан с учетом дребезга входного сигнала. Также для моделирования было изменено число счетчика дребезга для удобства моделирования. Результаты повторного моделирования представлены ниже, на рисунке 9.

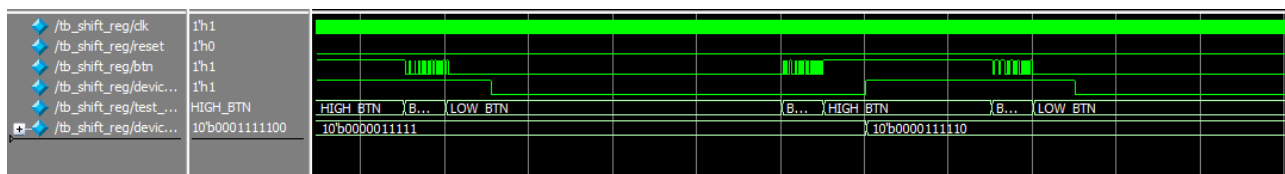


Рисунок 9 – Повторное моделирование с учетом дребезга

Как видно из повторного моделирования: схема устранения дребезга работает успешно и ложных срабатываний не происходит.

Проект был скомпилирован и загружен в ПЛИС. В ходе макетирования явления дребезга не наблюдалось: каждому нажатию кнопки соответствовал сдвиг числа на 1 бит.

### Выводы.

В ходе выполнения данной лабораторной работы были получены навыки создания проекта, его моделирования, имплементации в программируемую логическую интегральную схему (ПЛИС) с последующей внутрикристалльной отладкой проекта с использованием встроенного в САПР логического анализатора Signal Tap Logic Analyzer.