**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

2020

**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: EP4CE6E22C8. Подготовлено описание схемы в соответствии с вариантом задания, для чего был создан новый SV-файл. В основной модуль был подключён модуль вывода данных на семисегментные индикаторы, который применялся ранее в лабораторных работах.

Листинг 1. Исходный код описания схемы.

`define DIRECTION 1'b1 //Direction of shifting

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] semisigment\_digits, //Control semisigment digits

output logic [6:0] semisigment\_data //Displaying data

);

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

semisigments\_show show\_sems (

.i\_clock (clk),

.nreset\_i (reset),

.data\_dig\_3 (data\_reg[3:0]),

.data\_dig\_2 (data\_reg[7:4]),

.data\_dig\_1 ({2'b00, data\_reg[9:8]}),

.data\_dig\_0 (4'd0),

.digits (semisigment\_digits),

.lcd\_out (semisigment\_data)

);

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 ? {data\_reg[8:0], digit} : {digit, data\_reg[8:0]};

end

end

else

begin

prev\_btn = 1'b0;

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 3.

На рисунках 4 и 5 представлен RTL-вид проекта.

Расположение проекта в заданной ПЛИС представлено на рисунке 6.

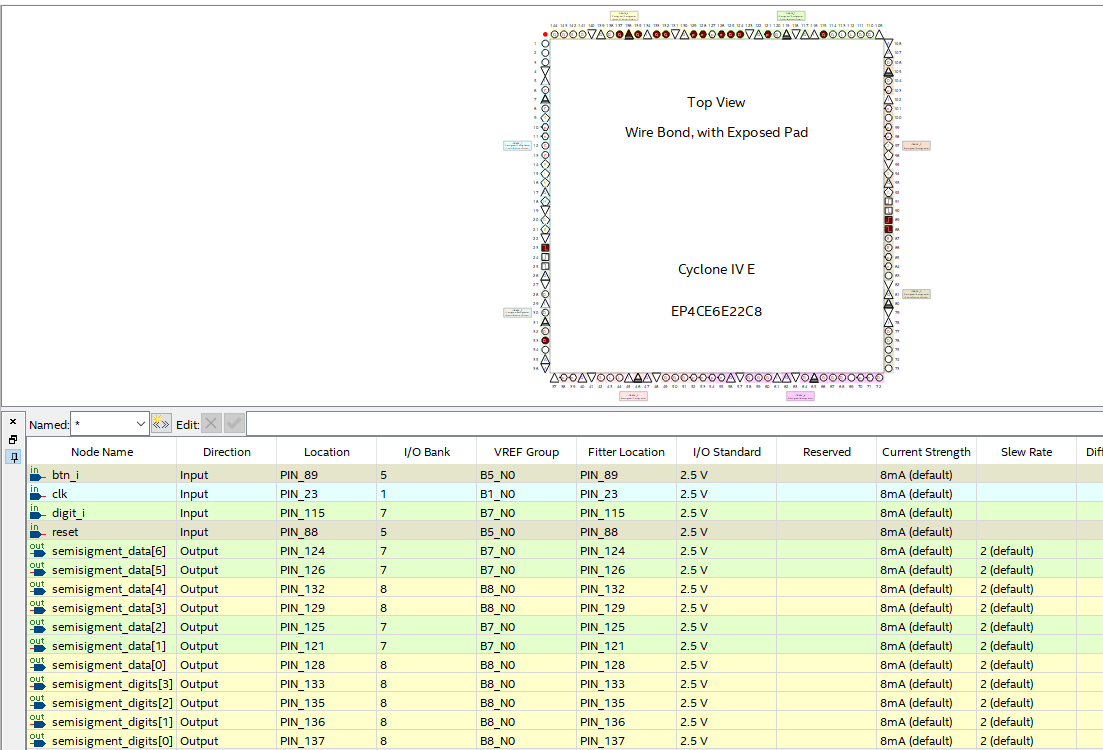


Рисунок 2 – Назначение входов и выходов проекта на контакты ПЛИС

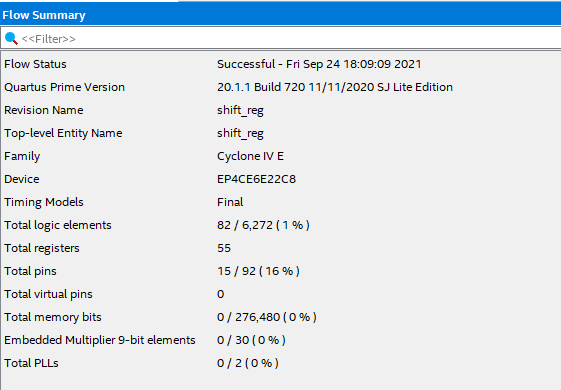


Рисунок 3 – Оценка затрат на реализацию проекта

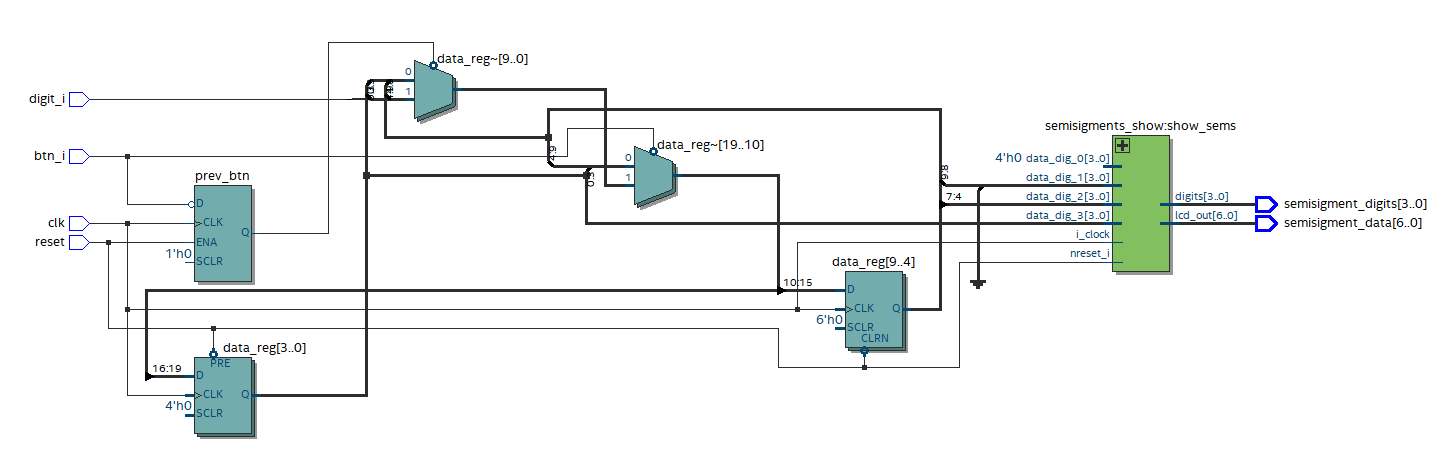


Рисунок 4 – RTL-представление проекта

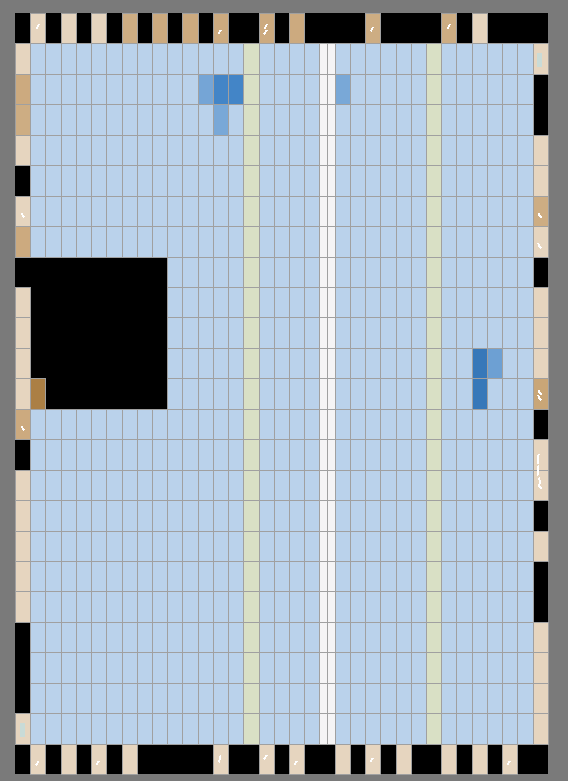


Рисунок 6 – Расположение проекта в заданной ПЛИС

***Этап 2. Моделирование проекта***

Для моделирования проекта был создан файл tb\_shift.reg, в котором были описаны входные возде йствия. Результаты моделирования представлены на рисунке \*.

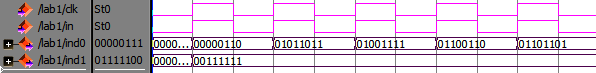


Рисунок 7 – Результат моделирования проекта

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-за дребезга контактов – каждый скачок происходит изменение значения данных, а таких скачков при нажатии кнопки непредсказуемое количество. Причем дребезг происходит как при нажатии, так и при отпускании кнопки.

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

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

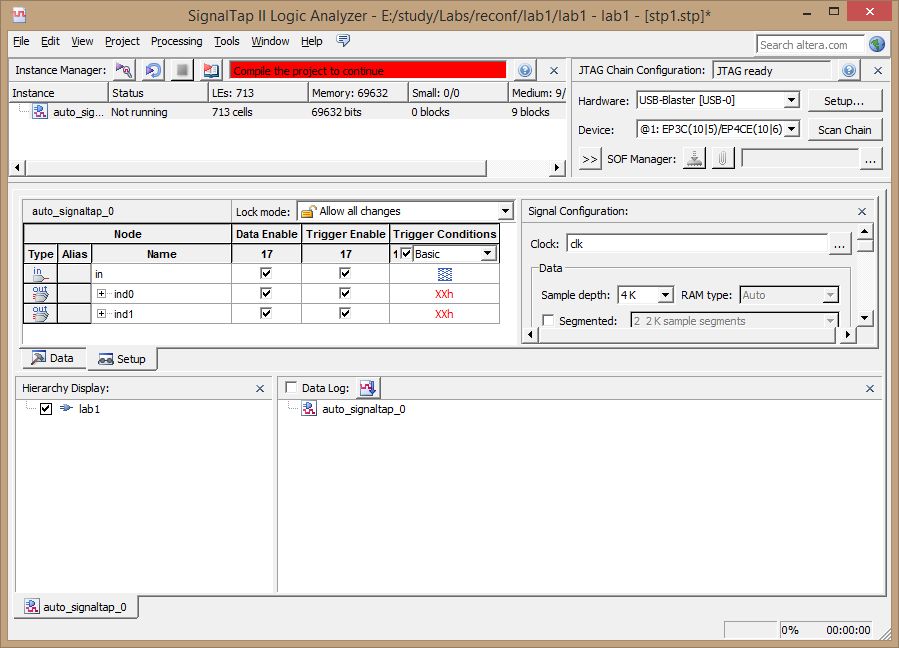


Рисунок 8 – Задание списка сигналов в Signal Tap II

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

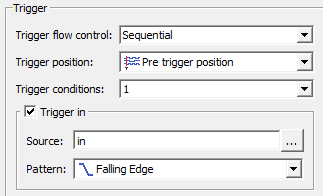


Рисунок 9 – Задание условия срабатывания логического анализатора

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



Рисунок 10 – 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 выполнено5 измерений времени дребезга.

Таблица 1. Измерение времени дребезга контактов.

|  |  |  |
| --- | --- | --- |
| N | Вр. диаграмма | Время дребезга, тактов clk |
| 1 |  | 138 |
| 2 |  | 27 |
| 3 |  | 2 |
| 4 |  | 71 |

Максимальное время дребезга составляет 138 тактов сигнала clk.

***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.

module vsm #(parameter time\_cnt\_len = 140)

(input clk, input in, output pulse);

parameter idle = 0;

parameter load\_count1 = 1;

parameter wait\_time1 = 2;

parameter wait\_sw = 3;

parameter load\_count2 = 4;

parameter wait\_time2 = 5;

parameter out\_pulse = 6;

reg [2:0] state = idle;

reg [2:0] time\_count = 0;

reg pulse\_reg = 0;

assign pulse = pulse\_reg;

always@ (posedge clk)

begin

case(state)

idle: begin

pulse\_reg <= 0;

if(in == 0)

state <= load\_count1;

end

load\_count1: begin

time\_count <= time\_cnt\_len;

state <= wait\_time1;

end

wait\_time1:

if(time\_count == 0)

state <= wait\_sw;

else

time\_count <= time\_count - 1;

wait\_sw:

if(in == 1)

state <= load\_count2;

else

state <= idle;

load\_count2: begin

time\_count <= time\_cnt\_len;

if(in == 1)

state <= wait\_time2;

else

state <= idle;

end

wait\_time2:

if(time\_count == 0)

state <= out\_pulse;

else

time\_count <= time\_count - 1;

out\_pulse: begin

pulse\_reg <= 1;

state <= idle;

end

endcase

end

endmodule

В исходном коде листинга 2 видно, что изначально автомат находится в состоянии idle, а значение сигнала pulse равно 0. Если в этом состоянии была нажата кнопка, автомат переходит в состояние load\_count1, в котором в счётчик загружается начальное значение, после чего автомат переходит в состояние wait\_time1, в котором значение счётчика уменьшается на 1, и находится в этом состоянии до тех пор, пока значение счётчика не станет нулевым. Когда значение счётчика стало нулевым, происходит переход в состояние wait\_sw. Если в состоянии wait\_sw оказалось, что кнопка всё ещё нажата, автомат переходит в состояние idle, иначе он переходит в состояние load\_count2, аналогичное load\_count1, но в котором тоже выполняется проверка нажатия кнопки и переход в состояния аналогично wait\_sw. Если кнопка была отпущена, происходит переход из состояния load\_count2 в состояние wait\_time2, аналогичное состоянию wait\_time1, но в котором при нулевом значении счётчика выполняется переход в состояние out\_pulse. В состоянии out\_pulse значение сигнала pulse устанавливается равным 1, далее происходит переход в состояние idle.

Диаграмма конечного автомата для устранения влияния дребезга контактов представлена на рисунке 11.

Проект был скомпилирован и загружен в ПЛИС. В ходе макетирования явления дребезга не наблюдалось: каждому нажатию кнопки соответствовало увеличение счётчика на 1.

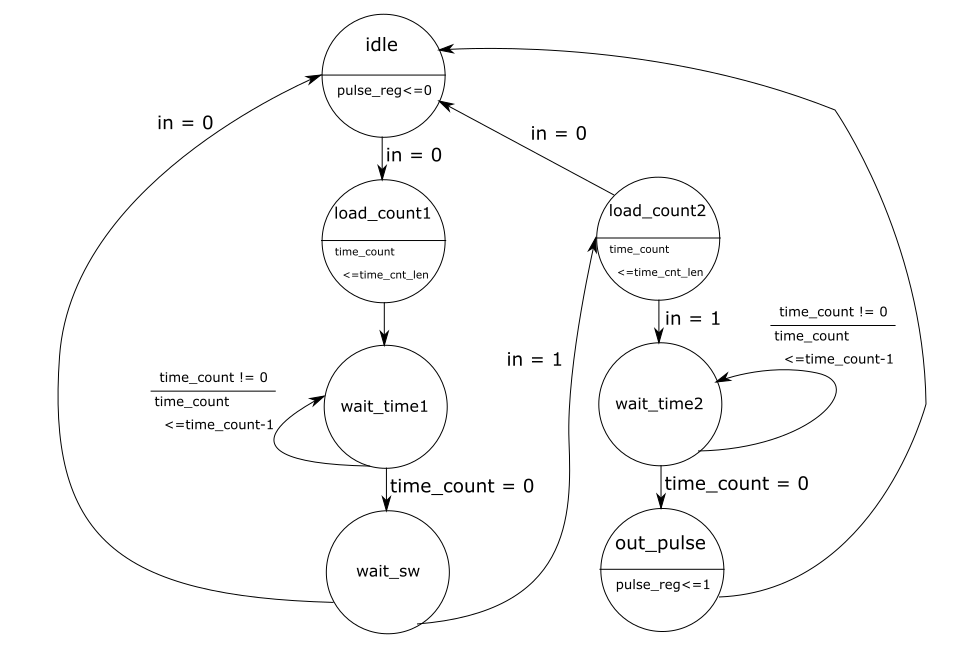


Рисунок – Диаграмма конечного автомата для устранения влияния дребезга  
контактов

**Выводы.**

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