Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
Московский авиационный институт
(национальный исследовательский университет)

Институт №3
«Системы управления, информатики и электроэнергетики»
Кафедра 304

Пояснительная записка к лабораторной работе №3
по учебной дисциплине «Автоматизация проектирования»
на тему: «Настройка системы программного процессорного ядра NiosII. Разработка системы управления движением на перекрёстке с использованием лабы DE2-115»

Выполнил:
студент группы М30-309Б-21
_____ Викулов Д. Г.

Руководитель работы:
_____ Ратников М. О.
«____»_____2023 г.

Москва 2023

Содержание

Цель работы

- Более глубокое понимание архитектуры микропроцессоров с программным ядром, а также архитектуры ЭВМ в целом;
- Изучение современных принципов проектирования с использованием Soft-микропроцессоров;
- Приобретение практических навыков проектирования микропроцессорных устройств:
- Приобретение навыков тестирования разработанных устройств
- Закрепление знаний по курсам «Встраиваемые вычислительные системы», «Компьютерная графика», «Операционные системы», «Программирование на языках высокого уровня».
- Освоение на базовом уровне QuartusII 10.0 WebEdition, SOPC_BUILDER, Eclipse C/C++
- Проектирование микропроцессорного устройства на базе ПЛИС.

Задание

Разработать программную часть для системы управления движением на перекрестке, согласно варианту 3 задания. Спроектировать микропроцессорное устройство на базе ПЛИС:

Вариант№	Система:	Дороги
3	Регулирования движением на перекрестке	1- главная (2 полоса), 2- второстепенная (1 полоса)

Спроектированная процессорная система

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Opcode Name
☑		⊟ **clk_0**	Clock Source						
		clk_in	Clock Input	**clk**					
		clk_in_reset	Reset Input	**reset**					
		clk	Clock Output	*Double-click to export*	clk_0				
		clk_reset	Reset Output	*Double-click to export*					
☑		⊟ **cpu_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]			IRQ 0	
		instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]			IRQ 31	
		jtag_debug_module_re...	Reset Output	*Double-click to export*	[clk]				
		jtag_debug_module	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_0800	0x0001_0fff		
		custom_instruction_m...	Custom Instruction Master	*Double-click to export*					
☑		⊟ **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_d9ff		
		reset1	Reset Input	*Double-click to export*	[clk1]				
☑		⊟ **switches**	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	**switches**					
☑		⊟ **LEDs**	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	**leds**					
☑		⊟ **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	0	

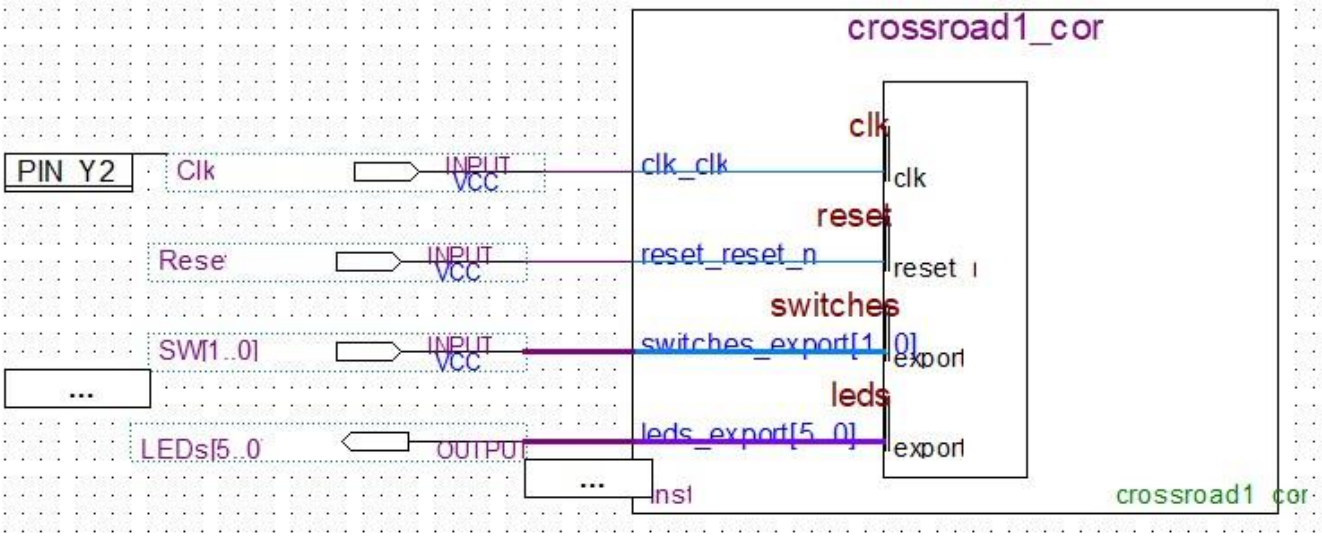
Карта адресов процессорной системы: (после генерации получаем файл crossroad1_core.sopcinfo, crossroad1_core.qsys)

	cpu_0.data_master	cpu_0.instruction_master
cpu_0.jtag_debug_module	0x0001_0800 - 0x0001_0fff	0x0001_0800 - 0x0001_0fff
onchip_memory2_0.s1	0x0000_8000 - 0x0000_d9ff	0x0000_8000 - 0x0000_d9ff
switches.s1	0x0001_1000 - 0x0001_100f	
LEDs.s1	0x0001_1010 - 0x0001_101f	
jtag_uart_0.avalon_jtag_slave	0x0001_1020 - 0x0001_1027	

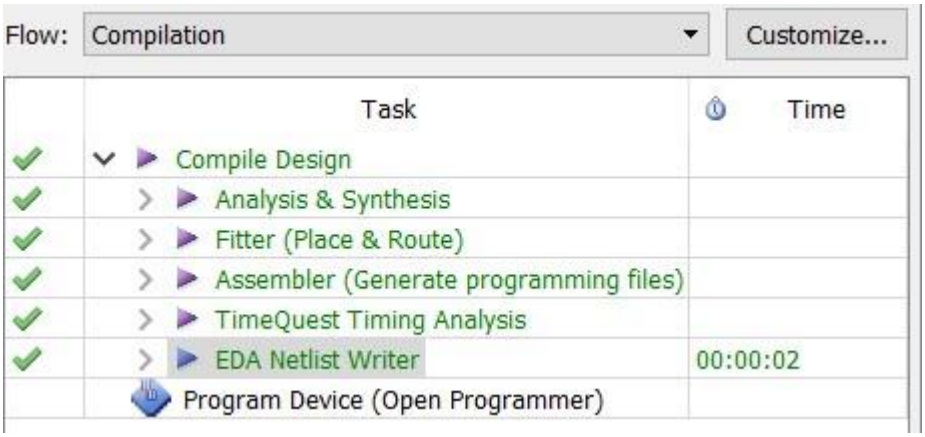
Файл верхнего уровня иерархии. Графический файл

Добавляем в проект процессорный модуль crossroad1_core. Получаем файл crossroad1_core.bsf

Создаём файл верхнего уровня иерархии:



Результаты полной компиляции:



Указываем этот файл в качестве верхнеуровневого, выбрав для него Set as Top-Level Entity.

Работа с контактами (настройка периферии)

Привязка контактов:

Named: * Edit: PIN_E22

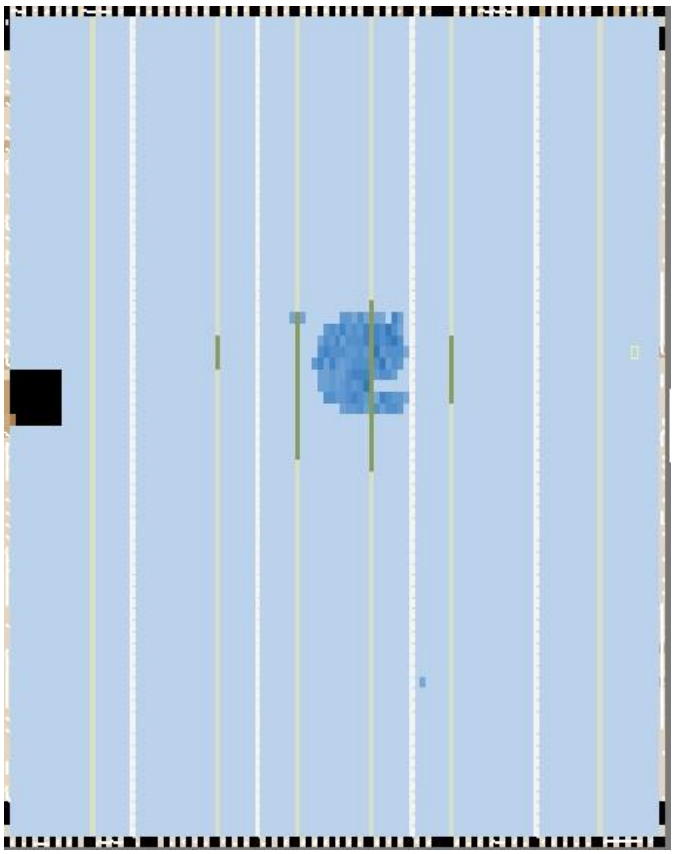
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
Clk	Input	PIN_Y2	2	B2_N0	PIN_Y2	2.5 V (default)		8mA (default)		
LEDs[5]	Output	PIN_E25	7	B7_N1	PIN_E25	2.5 V (default)		8mA (default)	2 (default)	
LEDs[4]	Output	PIN_E19	7	B7_N0	PIN_E19	2.5 V (default)		8mA (default)	2 (default)	
LEDs[3]	Output	PIN_E22	7	B7_N0	PIN_E22	2.5 V (default)		8mA (default)	2 (default)	
LEDs[2]	Output	PIN_F19	7	B7_N0	PIN_F19	2.5 V (default)		8mA (default)	2 (default)	
LEDs[1]	Output	PIN_E21	7	B7_N0	PIN_E21	2.5 V (default)		8mA (default)	2 (default)	
LEDs[0]	Output	PIN_G19	7	B7_N2	PIN_G19	2.5 V (default)		8mA (default)	2 (default)	
Reset	Input				PIN_Y1	2.5 V (default)		8mA (default)		
SW[1]	Input	PIN_AC28	5	B5_N2	PIN_AC28	2.5 V (default)		8mA (default)		
SW[0]	Input	PIN_AB28	5	B5_N1	PIN_AB28	2.5 V (default)		8mA (default)		
<<new node>>										

Назначение контактов:

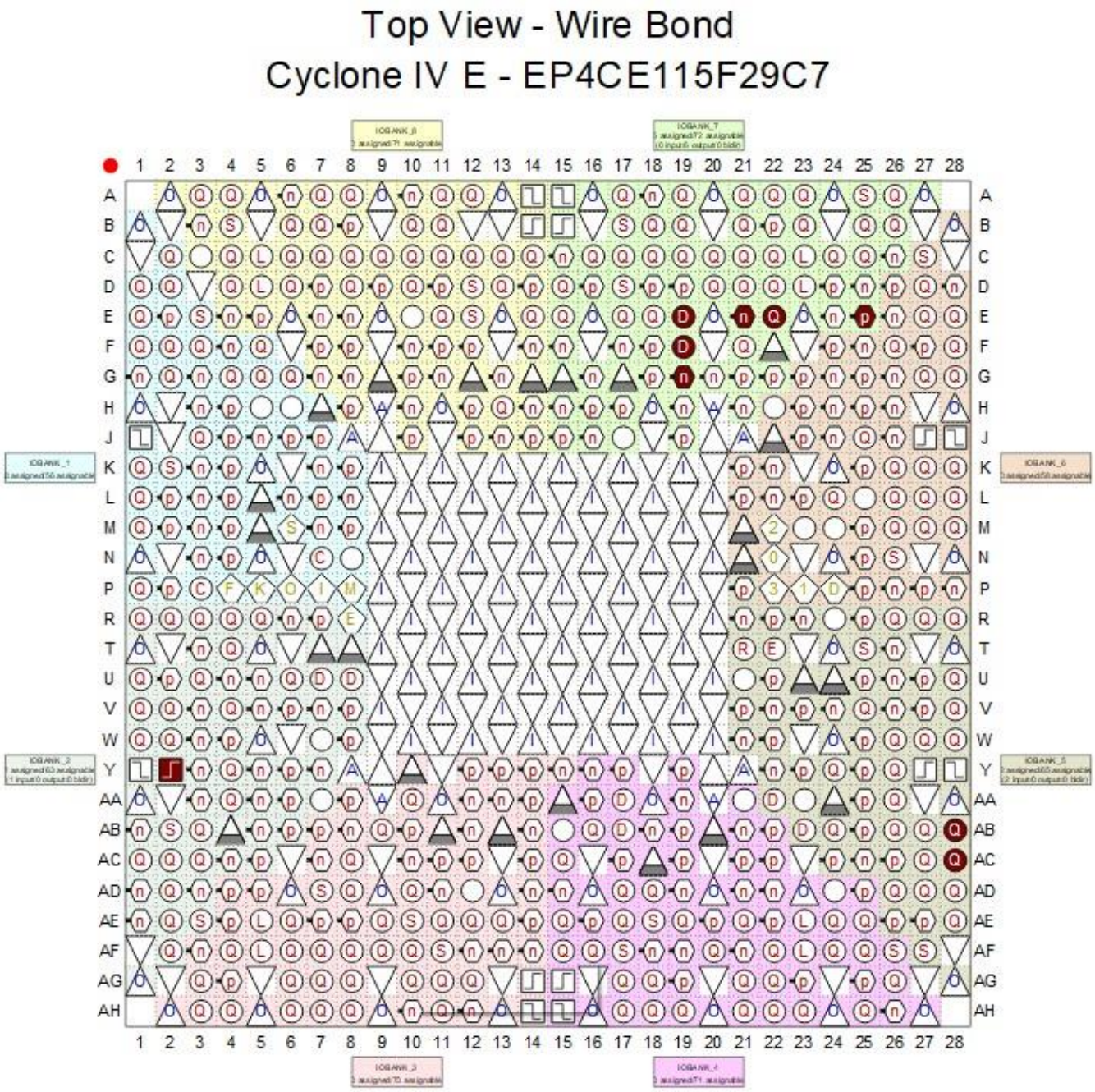
<<new>> Filter on node names: *

	Statu:	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	✓		Clk	Location	PIN_Y2	Yes			
2	✓		LEDs[5]	Location	PIN_E25	Yes			
3	✓		LEDs[4]	Location	PIN_E19	Yes			
4	✓		LEDs[3]	Location	PIN_E22	Yes			
5	✓		LEDs[2]	Location	PIN_F19	Yes			
6	✓		LEDs[1]	Location	PIN_E21	Yes			
7	✓		LEDs[0]	Location	PIN_G19	Yes			
8	✓		SW[1]	Location	PIN_AC28	Yes			
9	✓		SW[0]	Location	PIN_AB28	Yes			
10		<<new>>	<<new>>	<<new>>					

Прошивка в RTL Viewer:



Графическое изображение ПЛИС и назначенных контактов на ней:



Пины расставляются в соответствии со следующими инструкциями для платы:

Тумблеры:

Table 2: Pin assignments for slide switches

Signal Name	FPGA Pin No.	Description	I/O Standard
SW[0]	PIN_AB28	Slide Switch[0]	Depending on JP7
SW[1]	PIN_AC28	Slide Switch[1]	Depending on JP7
SW[2]	PIN_AC27	Slide Switch[2]	Depending on JP7
SW[3]	PIN_AD27	Slide Switch[3]	Depending on JP7
SW[4]	PIN_AB27	Slide Switch[4]	Depending on JP7
SW[5]	PIN_AC26	Slide Switch[5]	Depending on JP7
SW[6]	PIN_AD26	Slide Switch[6]	Depending on JP7
SW[7]	PIN_AB26	Slide Switch[7]	Depending on JP7
SW[8]	PIN_AC25	Slide Switch[8]	Depending on JP7
SW[9]	PIN_AB25	Slide Switch[9]	Depending on JP7
SW[10]	PIN_AC24	Slide Switch[10]	Depending on JP7
SW[11]	PIN_AB24	Slide Switch[11]	Depending on JP7
SW[12]	PIN_AB23	Slide Switch[12]	Depending on JP7
SW[13]	PIN_AA24	Slide Switch[13]	Depending on JP7
SW[14]	PIN_AA23	Slide Switch[14]	Depending on JP7
SW[15]	PIN_AA22	Slide Switch[15]	Depending on JP7
SW[16]	PIN_Y24	Slide Switch[16]	Depending on JP7
SW[17]	PIN_Y23	Slide Switch[17]	Depending on JP7

Привязка к светодиодам:

Table 4: Pin assignments for LEDs

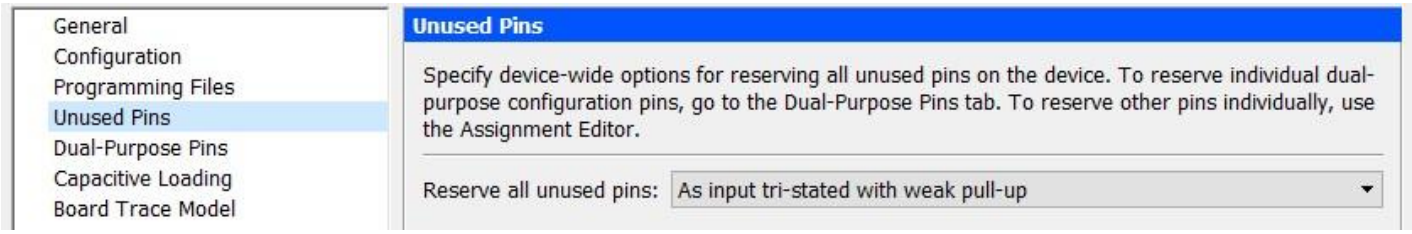
Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR[0]	PIN_G19	LED Red[0]	2.5V
LEDR[1]	PIN_F19	LED Red[1]	2.5V
LEDR[2]	PIN_E19	LED Red[2]	2.5V
LEDR[3]	PIN_F21	LED Red[3]	2.5V
LEDR[4]	PIN_F18	LED Red[4]	2.5V
LEDR[5]	PIN_E18	LED Red[5]	2.5V
LEDR[6]	PIN_J19	LED Red[6]	2.5V
LEDR[7]	PIN_H19	LED Red[7]	2.5V
LEDR[8]	PIN_J17	LED Red[8]	2.5V
LEDR[9]	PIN_G17	LED Red[9]	2.5V
LEDR[10]	PIN_J15	LED Red[10]	2.5V
LEDR[11]	PIN_H16	LED Red[11]	2.5V
LEDR[12]	PIN_J16	LED Red[12]	2.5V
LEDR[13]	PIN_H17	LED Red[13]	2.5V
LEDR[14]	PIN_F15	LED Red[14]	2.5V
LEDR[15]	PIN_G15	LED Red[15]	2.5V
LEDR[16]	PIN_G16	LED Red[16]	2.5V
LEDR[17]	PIN_H15	LED Red[17]	2.5V
LEDG[0]	PIN_E21	LED Green[0]	2.5V
LEDG[1]	PIN_E22	LED Green[1]	2.5V
LEDG[2]	PIN_E25	LED Green[2]	2.5V
LEDG[3]	PIN_E24	LED Green[3]	2.5V
LEDG[4]	PIN_H21	LED Green[4]	2.5V
LEDG[5]	PIN_G20	LED Green[5]	2.5V
LEDG[6]	PIN_G22	LED Green[6]	2.5V
LEDG[7]	PIN_G21	LED Green[7]	2.5V
LEDG[8]	PIN_F17	LED Green[8]	2.5V

Тактующий сигнал:

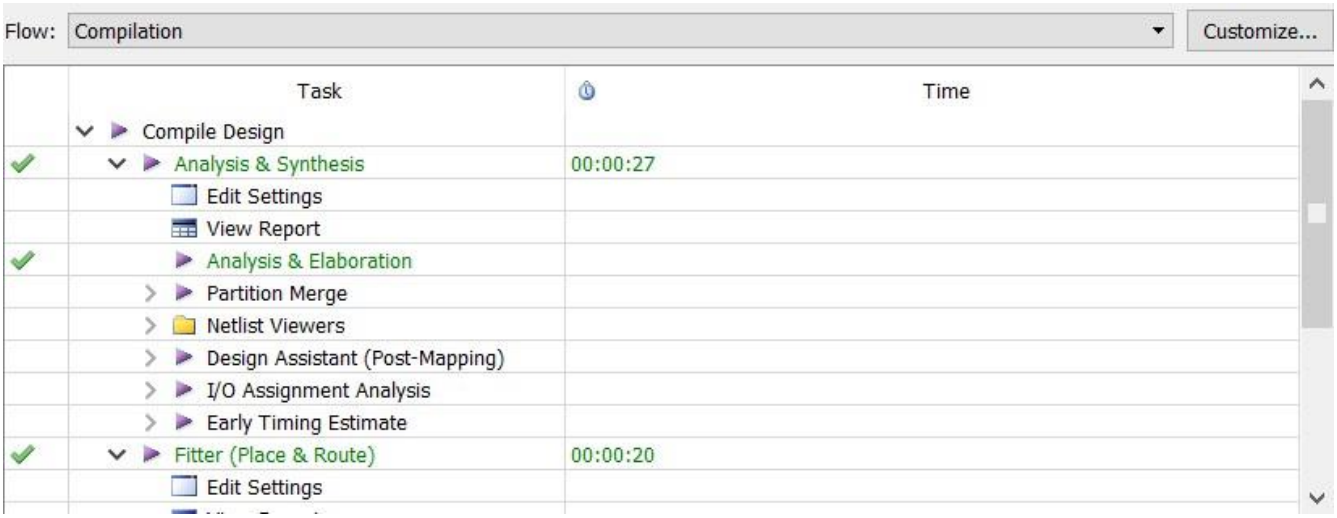
Table 6: Pin assignments for clock inputs

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
CLOCK_50	PIN_Y2	50 MHz clock input	3.3V
CLOCK2_50	PIN_AG14	50 MHz clock input	3.3V
CLOCK3_50	PIN_AG15	50 MHz clock input	Depending on JP6
SMA_CLKOUT	PIN_AE23	External (SMA) clock output	Depending on JP6
SMA_CLKIN	PIN_AH14	External (SMA) clock input	3.3V

Во избежание короткого замыкания при нахождении неиспользуемых пинов в состоянии «As output driving ground» переводим их в состояние «As input tri-stated with weak pull-up» - вывод со слабой подтяжкой к питанию (гарантируется напряжение высокого логического уровня до тех пор, пока оно не будет изменено извне каким-либо подключенным устройством).



Анализ и синтез схемы пройден успешно:



Получаем файл прошивки crossroad1.sof через выполнение Assembler из списка компиляции (Compile Design), который заливаем в ПЛИС.

Дальше дело за программной частью проекта.

Реализация программной части проекта с использованием Eclipse C/C++

Код программы:

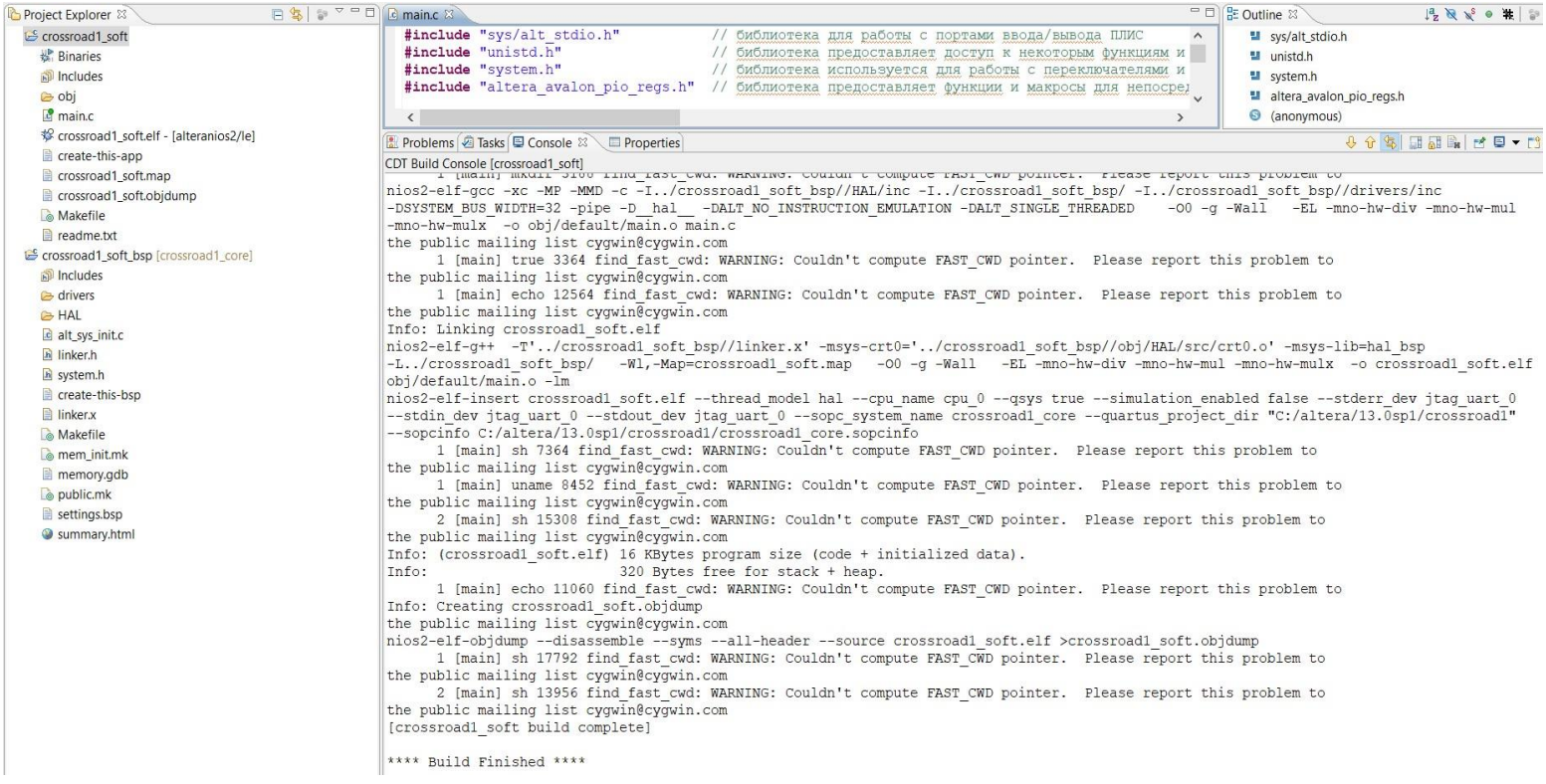
```
#include "sys/alt_stdio.h"       // библиотека для работы с портами ввода/вывода ПЛИС
#include "unistd.h"              // библиотека предоставляет доступ к некоторым
                                 // функциям и константам POSIX
#include "system.h"              // библиотека используется для работы с
                                 // переключателями и светодиодами, подключенными
                                 // к ядру PIO
#include "altera_avalon_pio_regs.h"  // библиотека предоставляет функции и макросы
                                     // для непосредственного чтения/записи
                                     // регистров ядра PIO

// определяем регистры ядра PIO
typedef struct
{
      // определяет состояние выходных портов ядра PIO. Эта переменная работает
      // как выходной буфер, который может быть использован для управления
      // внешними устройствами.
       unsigned long int DATA;

      // определяет направление данных на входных и выходных портах ядра PIO.
      // Если значение переменной DIRECTION установлено в 0, то порт
      // используется для вывода данных. Если значение установлено в 1, то порт
      //используется для ввода данных.
       unsigned long int DIRECTION;

      // определяет маску прерываний для соответствующего порта. Если
      // соответствующий бит установлен в 1, то прерывания от данного порта будут
      //запрещены. Если бит установлен в 0, то прерывания будут разрешены.
       unsigned long int INTERRUPT_MASK;

       // определяет обнаружение фронтов для каждого входного порта. Если флаг
       // установлен в 1, это означает, что произошло переключение на передний
       // фронт импульса на соответствующем пине порта
       unsigned long int EDGE_CAPTURE;
} PIO_STR;
```

```
// Макрос LEDS определяет указатель на структуру PIO_STR
// LEDS_BASE предполагается быть базовым адресом регистров ядра PIO
// Путем определения макроса LEDS, базовый адрес регистров приводится к типу
// указателя на структуру PIO_STR,
// что позволяет обращаться к регистрам ядра PIO через этот макрос.

// Макрос LEDS используется в коде для управления светодиодами на основе
// состояния переключателей, подключенных к ядру PIO
#define LEDS ((PIO_STR *) LEDS_BASE)


// ОСНОВНАЯ ФУНКЦИЯ
int main()
{
	// от состояния sw, код устанавливает режимы работы и состояние
	// светодиодов, подключенных к адресу LEDS_BASE интерфейса PIO
	int sw = -1;
	// значение используется для "самопереключения" состояний светофора
	int state = 0;

	// бесконечный цикл работы светофора
	while (1)
	{
		// проверка состояния переключателей. Считывание и сохранение в sw
		sw = IORD_ALTERA_AVALON_PIO_DATA(SWITCHES_BASE);

		// все задействованные LED переливаются – светодиоды
		// поочередно переключаются с интервалом 50 миллисекунд
		if (sw == 1)
		{
			LEDS->DATA = 1;
			usleep(50000);
			LEDS->DATA = 2;
			usleep(50000);
			LEDS->DATA = 4;
			usleep(50000);
			LEDS->DATA = 8;
			usleep(50000);
			LEDS->DATA = 16;
			usleep(50000);
			LEDS->DATA = 32;
```

```
        usleep(50000);
        // устанавливаем неопределённый режим работы до следующего
        // считывания состояния переключателей
        sw = -1;
}

// все задействованные светодиоды мигают, что имитирует перевод
// светофоров на мигающий режим работы с интервалом 300
// миллисекунд
else if(sw == 2)
{
                // светофоры все зажглись
                LEDS->DATA = 1 | 2 | 4 | 8 | 16 | 32;
                // подождали пока включены
                usleep(300000);
                // погасили светофоры
                LEDS->DATA = 0;
                // подержали их выключенными
                usleep(50000);
                // устанавливаем неопределённый режим работы до
                // следующего считывания состояния переключателей
                sw = -1;
}

 // иначе - нормальный режим работы светофоров перекрёстка. Они
 // самопроизвольно переключаются
 // включение определённых светодиодов в определённые
 // периоды времени. Значения комбинируются побитовой логикой
else
{
        if (state == 0)
        {
                LEDS->DATA = 1 | 4 | 32;
                usleep(2000000);
        }
        else if (state == 1)
        {
                LEDS->DATA = 2 | 8 | 16 ;
                usleep(1000000);
        }
        else if ( state == 2)
```

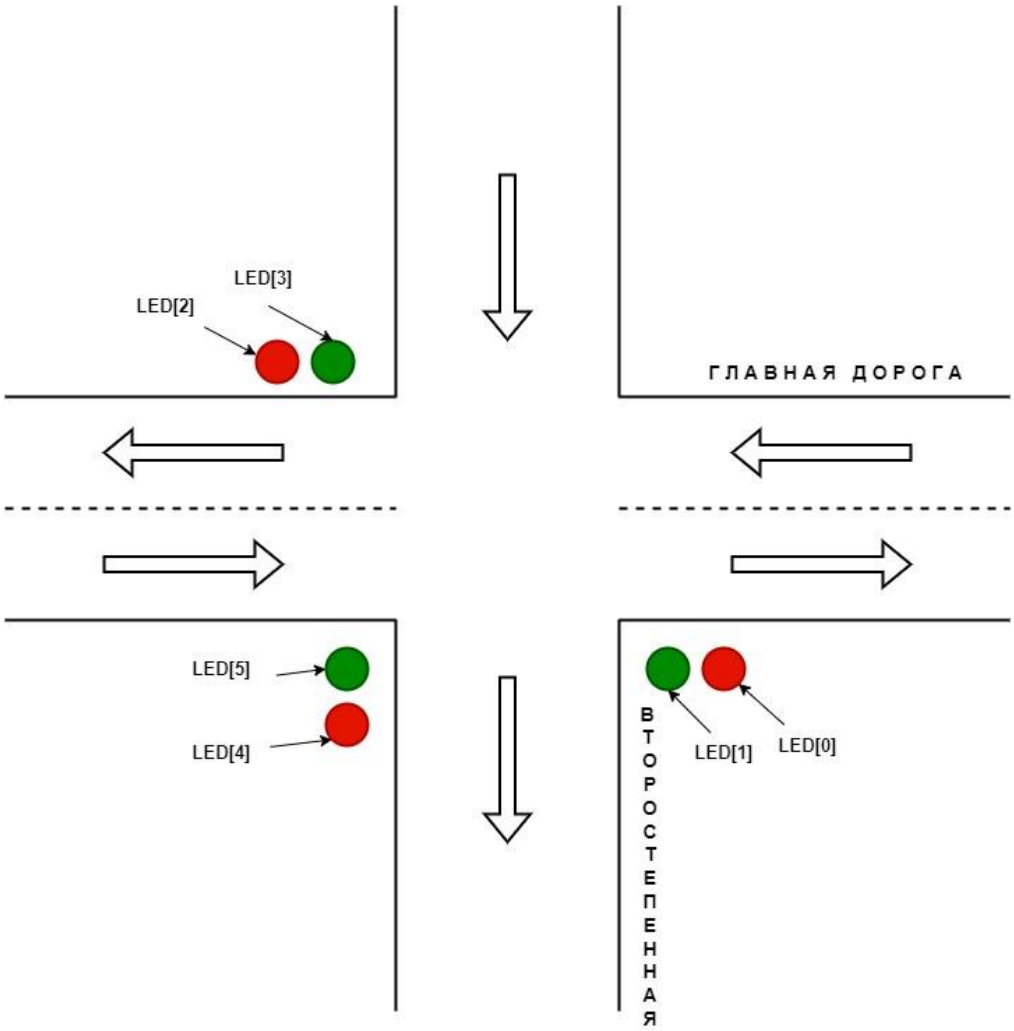
```
                {
                        LEDS->DATA = 1 | 8 | 16;
                        usleep(1500000);
                }
                else if (state == 3)
                {
                        LEDS->DATA = 2 | 4 | 16;
                        usleep(1500000);
                }
            }
            // переходим в следующее состояние по счётчику состояний
            state++;

            // сброс в начальное состояние после прохождения всего цикла
            // работы светофора
            if (state == 4)
            {
                state = 0;
            }
        }
    return 0;
}
```

После построения проекта через Build Project получаем сообщение в консоли об успешном завершении, и появляется необходимый файл crossroad1_soft.elf:



Тестирование программной части и работа с ПЛИС

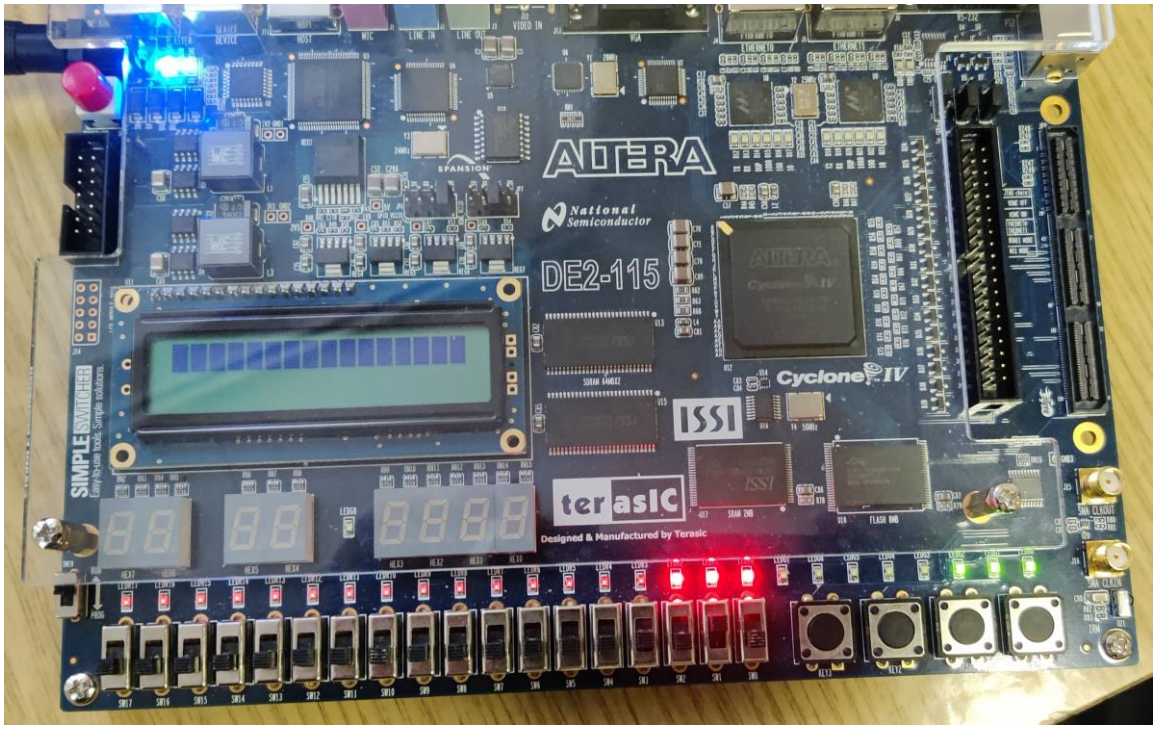
Расстановка светофоров и общий вид перекрестка с главной двухполосной дорогой и однополосной второстепенной выглядят следующим образом:



Общие принципы работы светофоров перекрёстка такого типа:

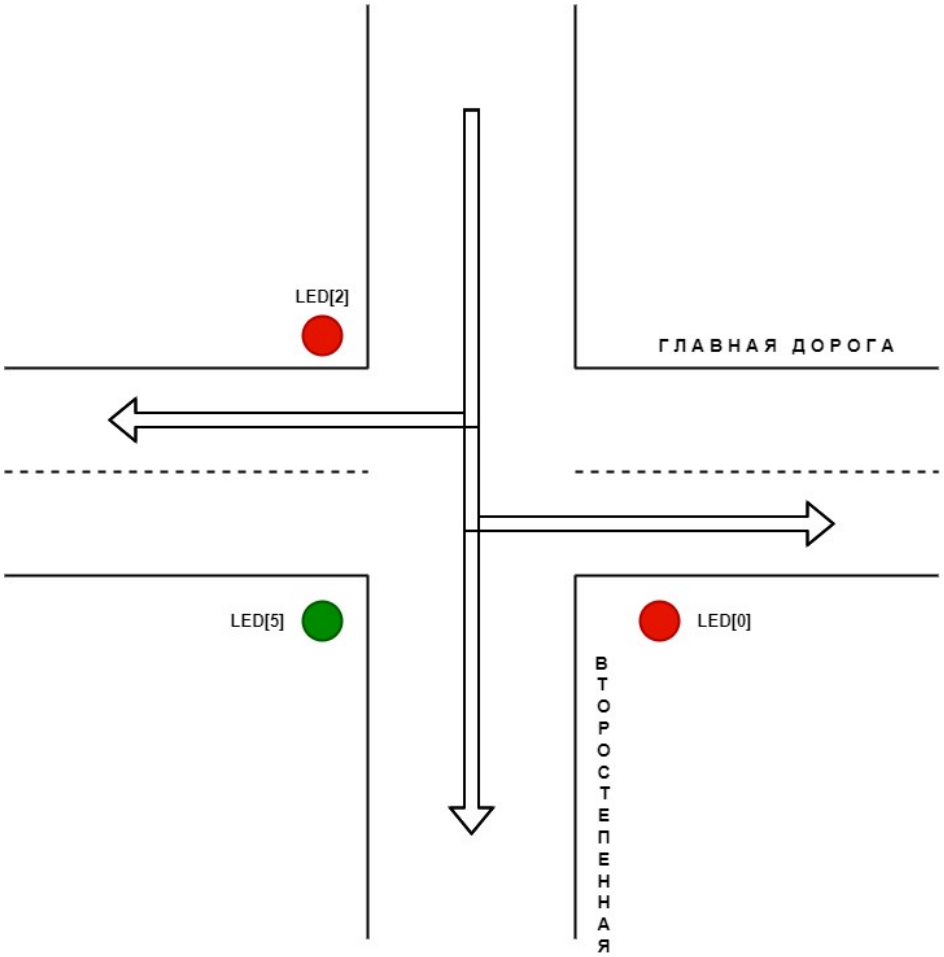
При нажатии на SW[0] все задействованные LED-диоды переливаются – светодиоды поочередно переключаются (светофоры сломаны). Картинку вставлять не имеет смысла – не отразит полную картину.

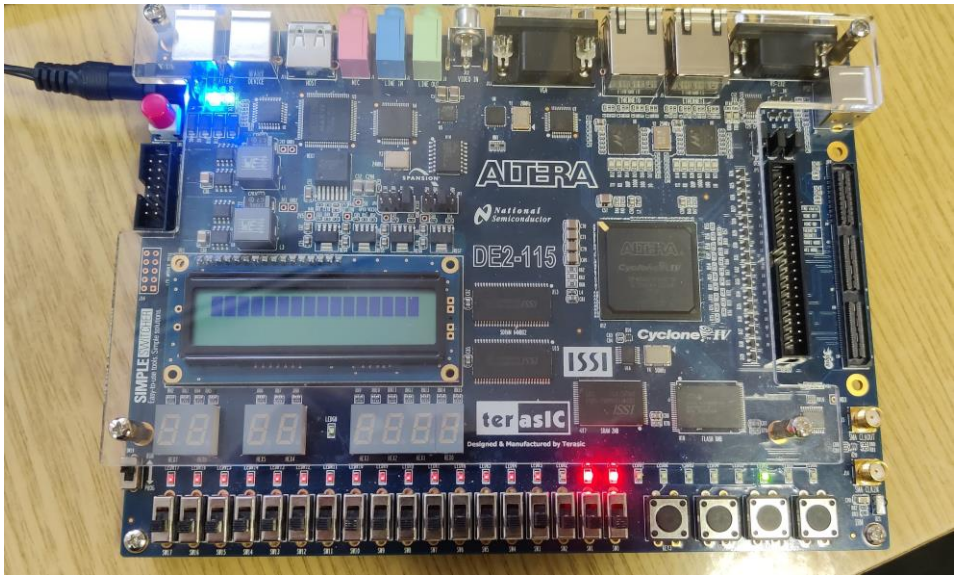
При нажатии на SW[1] все задействованные LED-диоды мигают, что имитирует перевод светофоров на мигающий режим работы (режим ожидания светофоров).



Дальше рассматриваем случаи переключения светофоров при функционировании в штатном режиме:

Состояние 1 при (SW=0 – ни один переключатель не включен):

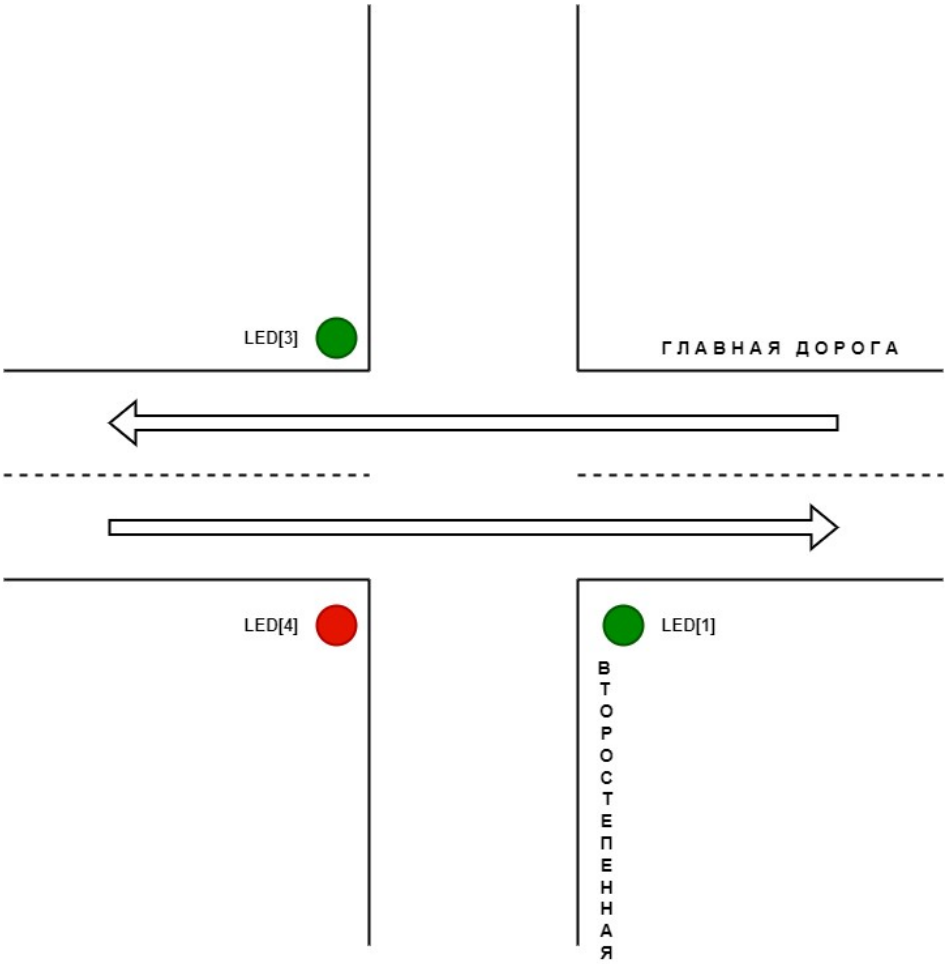


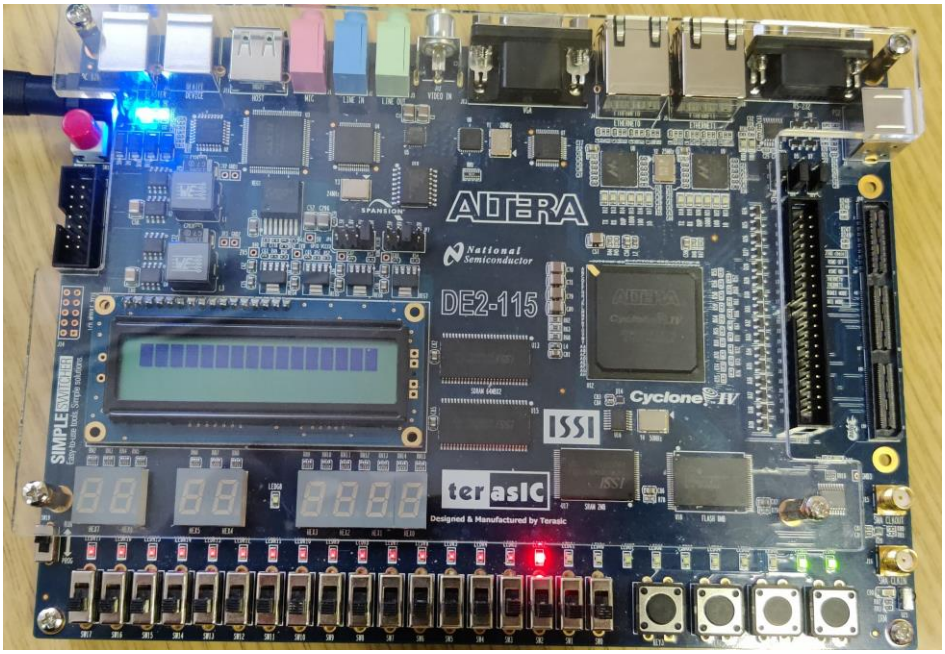


Машины едут только по второстепенной дороге (зелёный цвет светофора) с возможностью повернуть на главную дорогу (в этот момент по главной дороге машины не едут. Светофоры имеют красные цвета) в любое из направлений движения, согласующееся с правилами пересечения перекрёстка.

В состоянии 1 горят LED-диоды: LED[0], LED[2] и LED[5];

Состояние 2 при (SW=0 – ни один переключатель не включен):

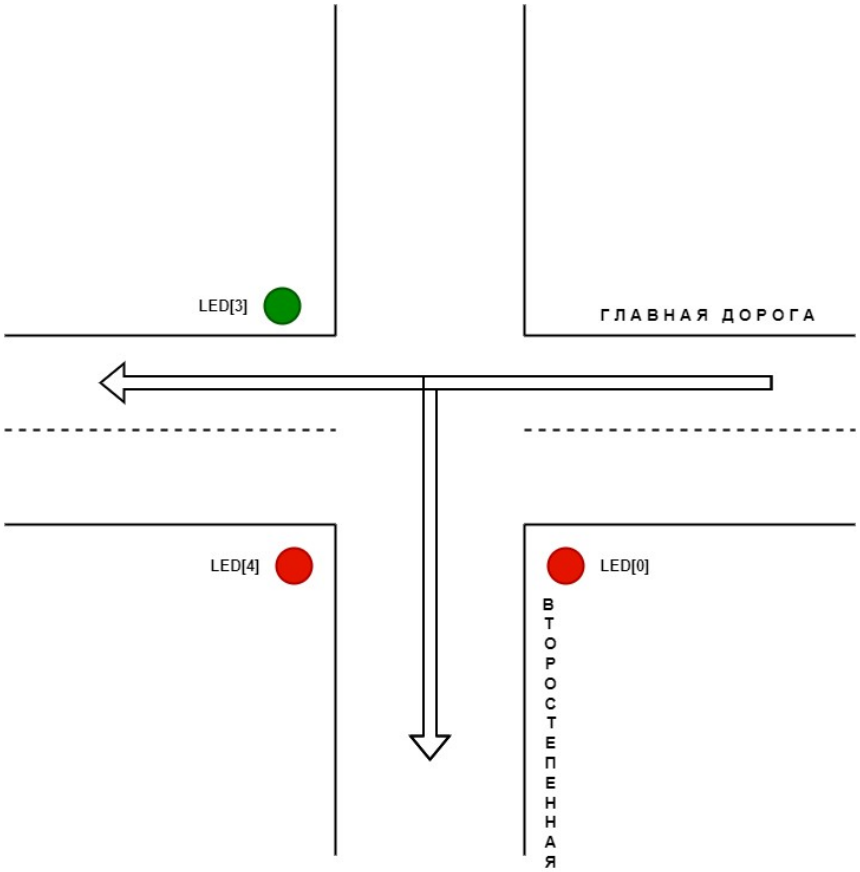


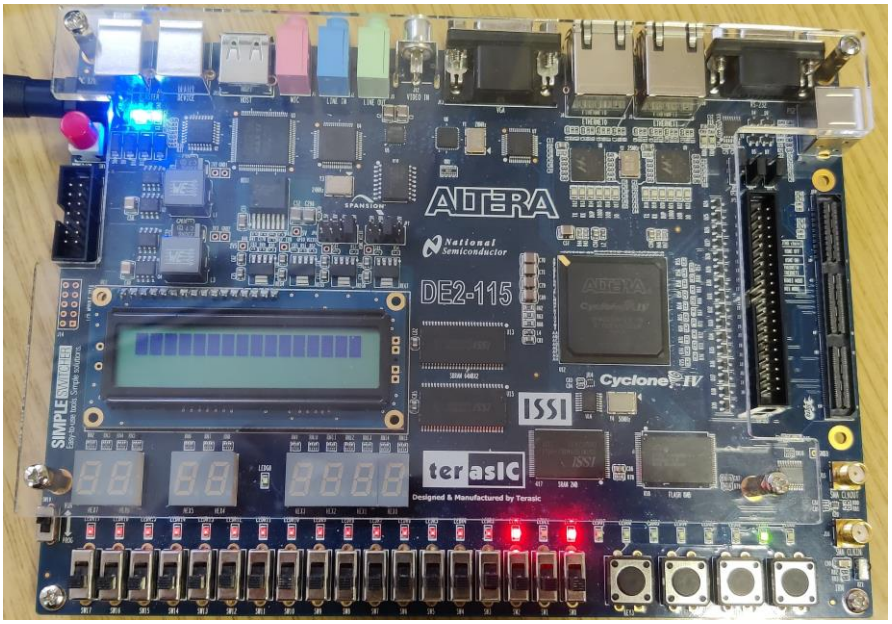


Машины едут только по главной дороге (два светофора имеют зелёные цвета) без возможности повернуть на второстепенную (красный цвет светофора), что согласуется с правилами пересечения перекрёстка.

В состоянии 2 горят LED-диоды: LED[1], LED[3] и LED[4];

Состояние 3 при (SW=0 – ни один переключатель не включен):

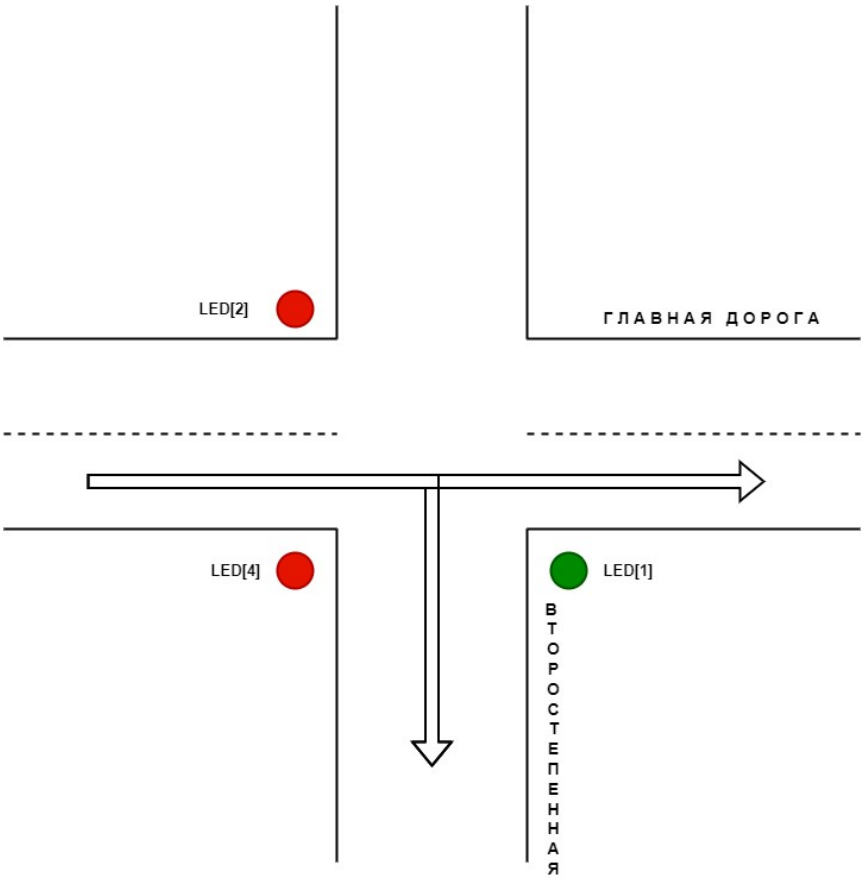


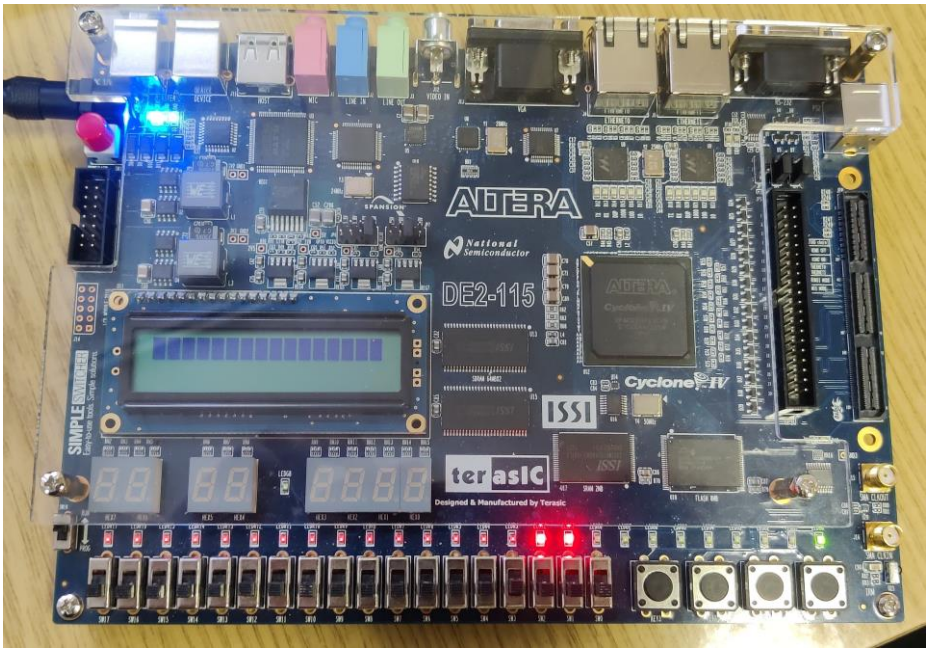


Машины поворачивают из полосы движения «влево» главной дороги на второстепенную или едут прямо. Причем светофор для движения машин, которые уже движутся по второстепенной дороге, переведён в красный цвет, и машины не движутся по другой полосе главной дороги (светофор тоже переведён в красный цвет), чтобы не допустить столкновения машин из двух полос.

В состоянии 3 горят LED-диоды: LED[0], LED[3] и LED[4];

Состояние 4 при (SW=0 – ни один переключатель не включен):





Машины поворачивают из полосы движения «вправо» главной дороги на второстепенную или едут прямо. Причем светофор для движения машин, которые уже движутся по второстепенной дороге, переведён в красный цвет и машины не движутся по другой полосе главной дороги (светофор тоже переведён в красный цвет), чтобы не допустить столкновения машин.

В состоянии 4 горят LED-диоды: LED[1], LED[2] и LED[4];

Вывод

В ходе выполнения лабораторной работы была разработана программа управления движением на перекрёстке для микросхемы типа DE2. Также были изучены принципы проектирования с использованием Soft-микропроцессоров, приобретены практические навыки проектирования микропроцессорных устройств: освоены на базовом уровне QuartusII 13.0sp1 WebEdition, SOPC_BUILDER, EclipseC/C++, и спроектировано микропроцессорное устройство на базе ПЛИС.

Список литературы

1. Работа №1. Нисходящее проектирование цифровых устройств на основе ПЛИС фирмы Altera. Создание проекта, описание устройства и его синтез, функциональное и временное моделирование, программирование ПЛИС. DOC-документ, прикреплённый в LMS ко второму номеру лабораторной работы курса «Элементы и узлы ЭВМ».

2. Лабораторная работа № 2. Лаб раб 230105 ВВС. DOC-документ, прикреплённый в LMS к третьему номеру лабораторной работы курса «Автоматизация проектирования».