1. ЛАБОРАТОРНАЯ РАБОТА №1

Тема работы: «Управление семисегментными индикаторами».

В лабораторной работе № 1 рассматриваются вопросы индикации при помощи семисегментных индикаторов, объединённых в дисплей в составе отладочной платы серии Xilinx/AMD Nexys [1].

Разрабатываемое устройство представляет собой устройство хранения истории ввода. Ввод очередного значения осуществляется при помощи движковых переключателей платы Xilinx/AMD Nexys A7. Для подтверждения ввода используется кнопка BTN_C платы Xilinx/AMD Nexys A7. Хранимая история ввода должна отображаться при помощи семисегментных индикаторов в составе платы Xilinx/AMD Nexys A7 [1]. Последнее введённое значение должно отображаться на крайнем правом индикаторе правого дисплея. Далее, справа налево должны располагаться ранее введённые значения в порядке их прихода (чем раньше пришло значение, тем левее оно располагается). Количество хранимых значений прямо пропорционально количеству индикаторов. Для отладочной платы Xilinx/AMD Nexys A7 число хранимых значений равно восьми.

Задание на лабораторную работу.

1. Создать проект в САПР Vivado для ПЛИС Artix-7 **xc7a100tcsg324-1I** [1].
2. Создать модуль управления семисегментными индикаторами. Предусмотреть возможность отключения части семисегментных индикаторов посредством задания маски.
3. Создать модуль верхнего уровня.

3.1. Создать сдвиговый регистр для хранения истории ввода.

3.2. Обеспечить подачу на вход регистра данных с движковых переключателей.

3.3. Реализовать схему разрешения записи сигнала при подаче управляющего воздействия через кнопку (через фильтр дребезга контактов).

3.4. Реализовать запись в регистр согласно постановке задачи.

3.5. Обеспечить вывод информации с регистра на семисегментные индикаторы.

4. Создать тестовый модуль на языке Verilog HDL. Протестировать описанную ранее схему.
5. Добавить файл проектных ограничений в проект. Обеспечить связь входов и выходов схемы с портами ПЛИС.
6. Сгенерировать файл с расширением «.bit». Загрузить файл на плату.
7. Произвести верификацию.
8. Составить отчёт.

1.1. Теоретическое введение

1.1.1. Семисегментный индикатор

Светодиод является одним из наиболее простых устройств индикации. Светодиод, являясь диодом, имеет в своём составе такие структурные элементы, как анод и катод. Обозначение светодиода на электрических схемах приведено на рис. 1.1.

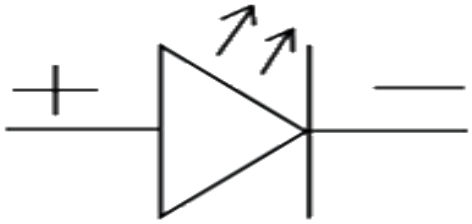

Рисунок 1.1. Обозначение светодиода на электрических схемах

В случае, если корпус светодиода выпуклый, он работает, как собирающая линза, если плоский, то как рассеивающая. Светодиод, который рассматривается при работе с платой, представляет собой SMD компонент.

Схематичное изображение светодиода с выпуклым корпусом представлено на рис. 1.2.

Схематичное изображение SMD компонента светодиода представлено на рис. 1.3.

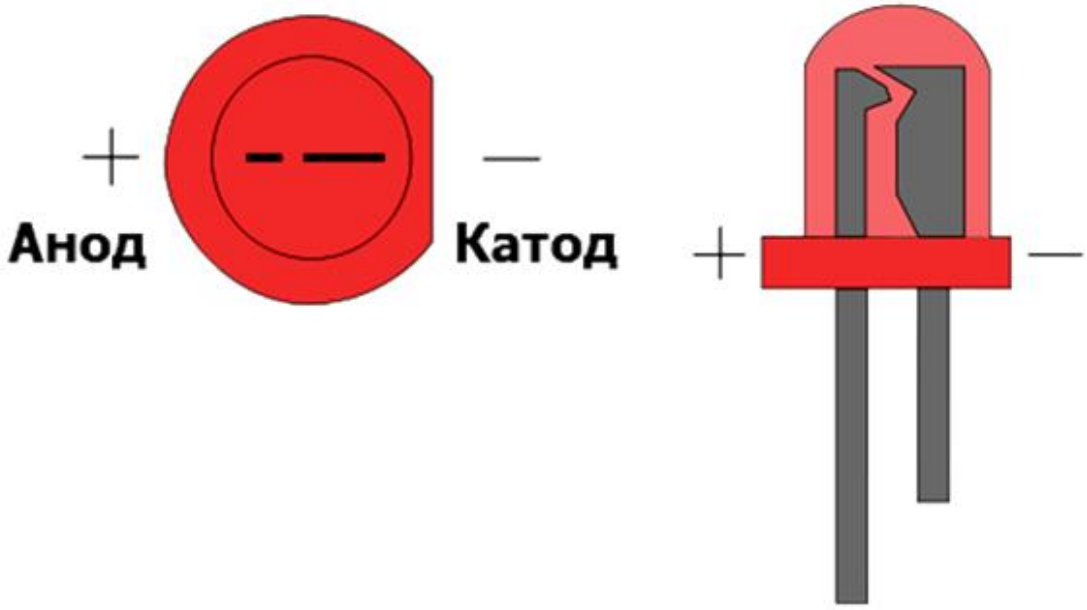

Рисунок 1.2. Схематичное отображение светодиода виды сверху и сбоку

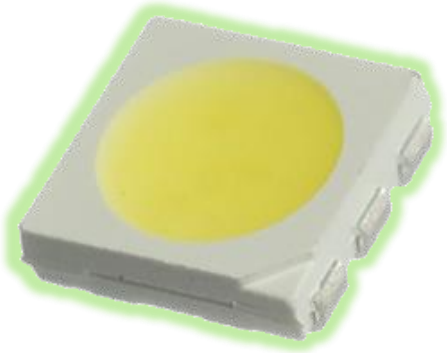

Рисунок 1.3. Схематичное изображение SMD компонента светодиода

Наличие у светодиода свойства светится определяется составом материалов, из которых светодиод создаётся. Цвет светодиода точно также зависит от материалов. Основной принцип свечения состоит в том, что при рекомбинации дырок и электронов энергия выходит в виде фотонов, как следствие, излучается свет.

Невозможно физически сделать светодиод, который будет светиться белым светом. Для получения белого свечения используют либо принцип RGB (цвет получается не совсем белым, а близко к белому), либо цвет корректируется при помощи люминофора.

Инфракрасное: напряжение — до 0,7 В. Белый цвет — от 3,7 В.

В основе семисегментного индикатора лежит набор светодиодов. Они выстроены определённым образом для повторения очертаний символов букв и цифр.

На отладочной плате присутствуют два блока по четыре семисегментных индикатора. Внешний вид такого дисплея приведён на рис. 1.4.

Схема семисегментного индикатора приведена на рис. 1.5. Для семисегментного индикатора характерно 7 или 8 светодиодов (на рисунке представлена конфигурация с «точкой»). Все светодиоды индикатора проименованы по часовой стрелке буквами от A до G. Соответственно, для отображения информации при помощи такого индикатора необходимо зажечь часть светодиодов в соответствии с желаемым символом.

Рисунок 1.4. Схема семисегментного индикатора

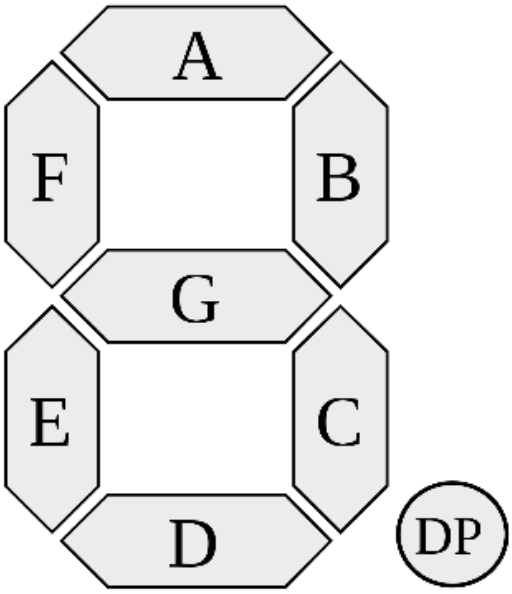

Рисунок 1.5. Схема семисегментного индикатора

Индикатор состоит из 7 сегментов, каждый из которых представляет собой светодиод. Также присутствует отдельный светодиод для обозначения точки. Для управления таким индикатором необходимо подать сигнал на каждый светодиод в его составе. При этом те светодиоды, на которые будет подан нужный сигнал загорятся соответствующим цветом (красным).

Все светодиоды в рамках одного индикатора имеют один общий анод, при этом катод у каждого свой (рис. 1.6). Для формирования конкретного символа следует подать высокий уровень сигнала на общий анод, а также соединить с шиной земли (низкий уровень) катоды тех светодиодов, которые должны зажечься для формирования символа.

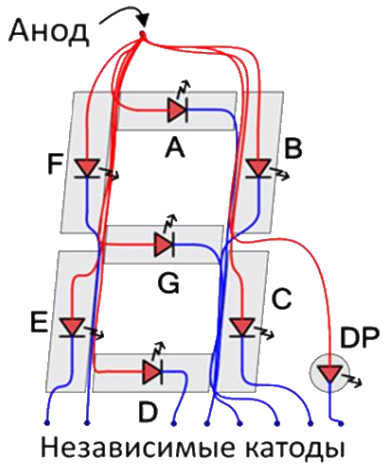

Рисунок 1.6. Схема соединения анодов и катодов в рамках индикатора по схеме с общим анодом

Далее стоит рассмотреть устройство одного блока семисегментных индикаторов (рис. 1.7).

Рисунок 1.7. Схематичное представление блока семисегментных индикаторов

У каждого индикатора присутствует свой отдельный анод, а вот с катодами ситуация обстоит сложнее: катоды идентичных светодиодов (обозначенных одной и той же буквой) разных индикаторов собраны в общую шину (рис. 1.8).

Вследствие этой конструктивной особенности в один и тот же дискретный момент времени на всех индикаторах в пределах одного блока может быть отображена только одна и та же информация (одинаковые символы). Встаёт вопрос о том, как организовать вывод информации таким образом, чтобы иметь возможность отобразить различные символы в пределах одного блока. Для этого используется принцип динамической индикации.

Эффект основывается на том, что человек способен различать разные изображения только с частотой ниже определённой границы. Поэтому суть отображения многоразрядного числа сводится к последовательному отображению каждого из его разрядов с большой частотой.

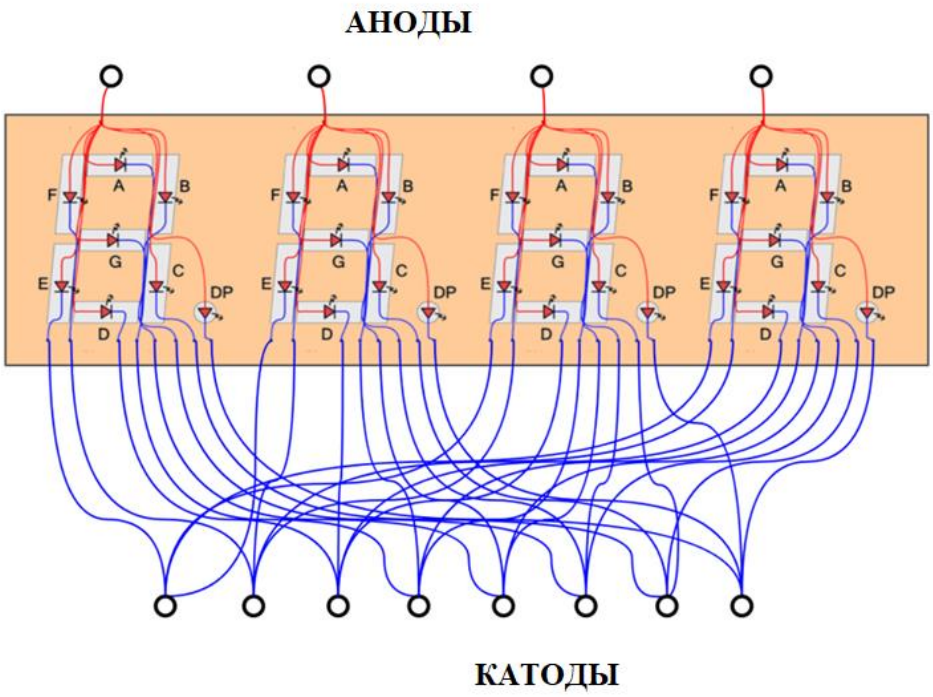

Рисунок 1.8. Схематичное представление соединений анодов и катодов внутри блока семисегментных индикаторов

При достаточно высокой частоте изменения изображения оно будет сливаться в единую картинку (в силу биологических особенностей человека). При этом следует учесть, что если частота будет слишком большая, то моменты, когда индикатор не горит, будут появляться больше за фиксированное время, чем при меньшей частоте, а значит глаз человека будет воспринимать изображение, как более тёмное. При проектировании важно подобрать частоту, комфортную для восприятия человеком; частота подбирается экспериментально.

Далее стоит рассмотреть схему подключения семисегментных дисплеев на отладочной плате серии Xilinx/AMD Nexys [1] (рис. 1.9).

В силу особенностей подключения все индикаторы на обоих дисплеях имеют общие катоды (по тому же принципу, что объяснялось ранее). Соответственно, принцип динамической индикации должен распространяться поочерёдно на два дисплея.

Для управления анодами на вход каждого анода установлен транзистор. В рамках данного повествования транзистор рассматривается, как управляемый переключатель, не вдаваясь в физические особенности процессов, происходящих при переключении. При подаче низкого уровня на переключатель на соответствующий анод подаётся высокий уровень сигнала, в противном случае – низкий уровень сигнала.

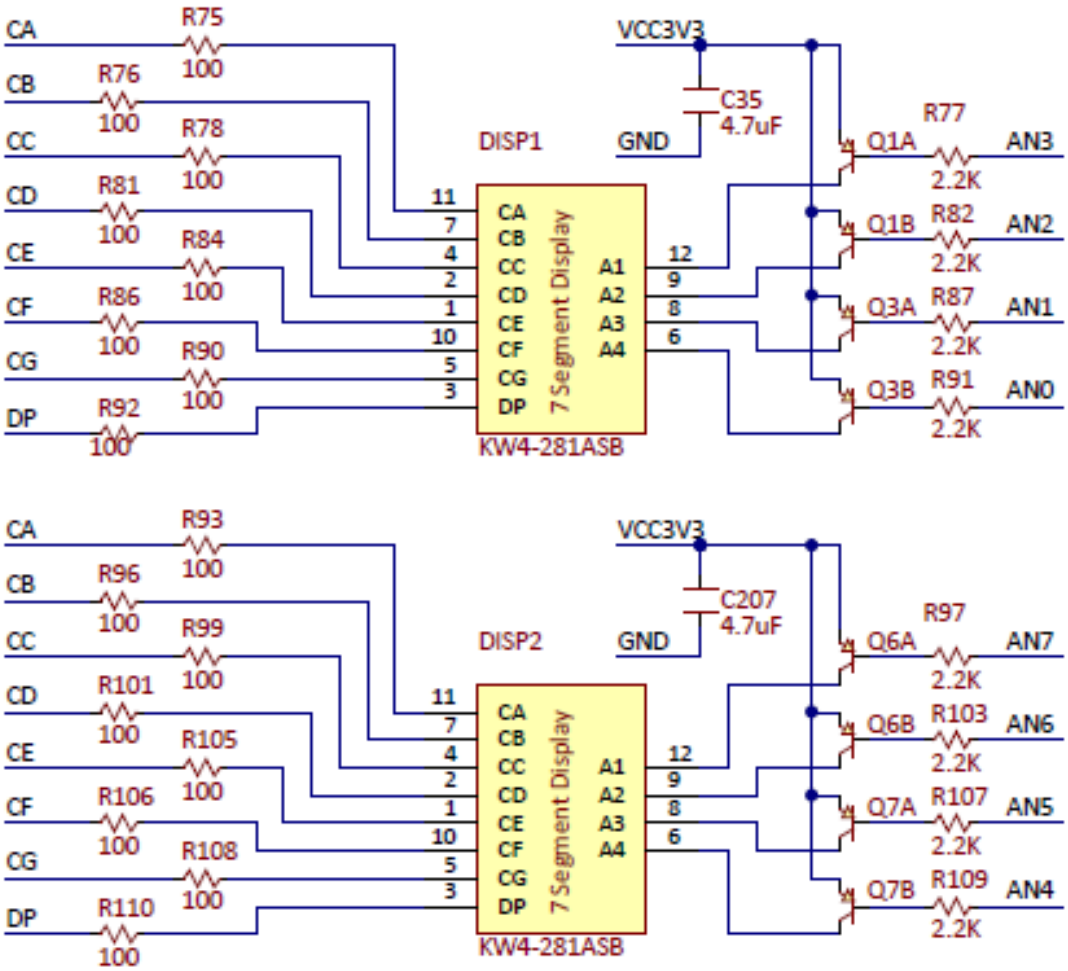

Рисунок 1.9. Схема подключения семисегментных дисплеев на отладочной плате серии Xilinx/AMD Nexys

Соответственно, при реализации устройства управления семисегментными индикаторами необходимо учитывать частоту обновления (делитель частоты) и принцип управления светодиодами на плате.

1.1.2. Кнопка

Кнопка — устройство, представляющее собой физический датчик, который фиксирует событие физического взаимодействия с ним и замыкает или размыкает электрическую цепь, к которой он подключен.

Внешний вид используемой кнопки представлен на рис. 1.10.

На плате Xilinx/AMD Nexys A7 [1] присутствует пять кнопок, они расположены так, как показано на рис. 1.11. Также присутствуют две кнопки красного цвета, отличающиеся по принципу работы.

Рисунок 1.10. Внешний вид кнопки

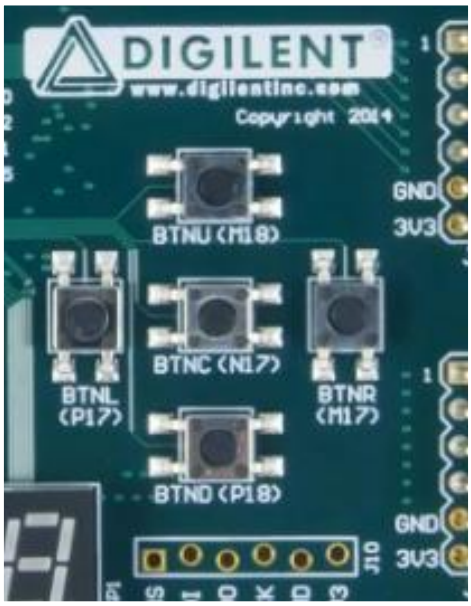

Рисунок 1.11. Расположение кнопок на отладочной плате серии Xilinx/AMD Nexys

Схема подключения кнопок представлена на рис. 1.12.

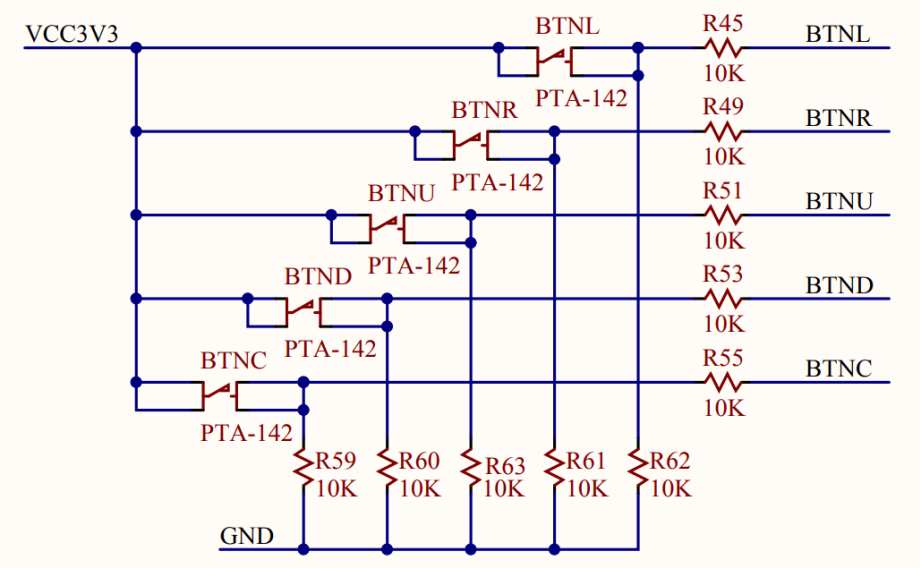

Рисунок 1.12. Схема подключения кнопок на отладочной плате серии Xilinx/AMD Nexys

Согласно схеме подключения, каждая из кнопок связана pull-down резистором. Подтягивающий резистор нужен для обеспечения стабильного сигнала при разомкнутом контакте. Pull-down резисторы обеспечивают подтяжку к земле (резисторы R59-R62).

Аналогичная ситуация с Pull-up резистором, однако подтяжка происходит к линии питания (VCC). Таким образом подключены кнопки «CPU_RESET» и «PROG» (рис. 1.13).

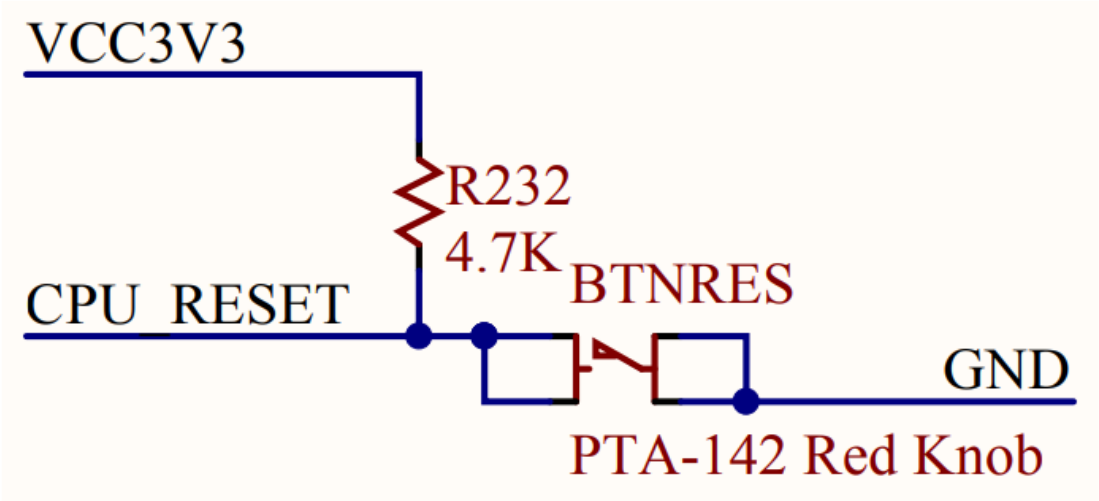

Рисунок 1.13. Схема подключения кнопок «CPU_RESET» и «PROG» на отладочной плате серии Xilinx/AMD Nexys

При эксплуатации тактируемых систем может возникнуть следующая ситуация: пусть в схеме присутствует D-триггер синхронный динамический. Такой триггер может изменять своё состояние только в момент прихода фронта синхросигнала. В случае, если сигнал на информационном входе триггера появляется с соблюдением необходимых временных задержек (предустановка и выдержка с точки зрения временных интервалов), то триггер корректно перейдёт в новое состояние.

Далее стоит рассмотреть ситуацию, при которой сигнал на входе триггера формируется асинхронно (например, человек жмёт на кнопку). Такой сигнал может возникнуть в абсолютно произвольный момент времени, а значит может совпасть по времени с моментом, когда синхросигнал перейдёт в такое состояние, которое запретит триггеру откликаться на изменения на его информационном входе. Тогда процесс перехода триггера в новое состояние может начаться, а затем прекратиться в некотором промежуточном состоянии.

Спустя некоторое время триггер при отсутствии внешних воздействий придёт к одному из двух своих стабильных состояний, однако, если остановка перехода произошла в момент времени, крайне близкий к середине процесса

(равновесное состояние), то период, пока триггер определится куда ему стабилизироваться, может занять долгое время, что приведёт к непредсказуемой реакции оставшейся части системы. Подобные аномалии называются метастабильными состояниями. Метастабильность невозможно побороть. Можно лишь уменьшить возможность появления метастабильности, используя синхронизаторы.

Одним из самых распространенных синхронизаторов является синхронизатор ждущего типа («brute-force») [2]. Схема синхронизатора такого типа приведена на рис. 1.14.

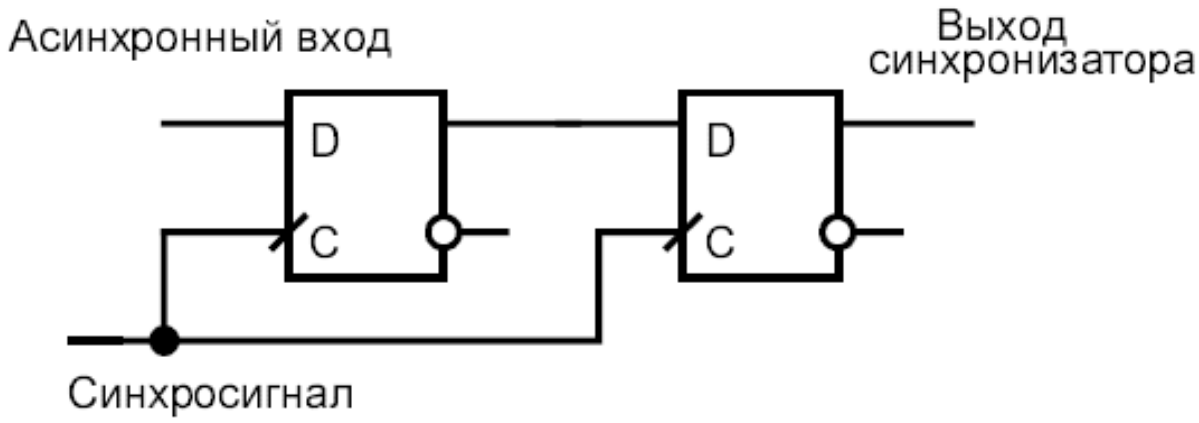

Рисунок 1.14. Схема синхронизатора ждущего типа

Состоит он из нескольких последовательно соединенных друг с другом триггеров.

Первый триггер принимает асинхронный сигнал и, соответственно, при ситуации, описанной выше, попадает в метастабильное состояние. Второй триггер откликается на сигнал с первого триггера лишь по прошествии такта, а за это время большинство коротких метастабильностей успеет затухнуть. Чем больше триггеров будет в цепочке, тем меньше вероятность аномалий. Соответственно нужно понимать, что сигнал при такой схеме попадёт на основное обрабатывающее устройство спустя столько тактов, сколько триггеров будет в цепочке.

При манипуляции кнопкой с помощью механического воздействия наблюдается такой эффект, как дребезг контактов [2, 3].

Суть его заключается в том, что за счёт несовершенства физического воплощения кнопки, а точнее, проводящих металлических пластин, в момент нажатия или отжатия кнопки протекающие физические процессы не позволяют мгновенно установить активный сигнал на выходе. Некоторое (очень короткое)

время протекания эти процессов сигнал на выходе будет изменяться от нуля до единицы и обратно (рис. 1.15). Такой эффект затрудняет регистрацию значения с выхода кнопки и требует дополнительной обработки при помощи фильтра.

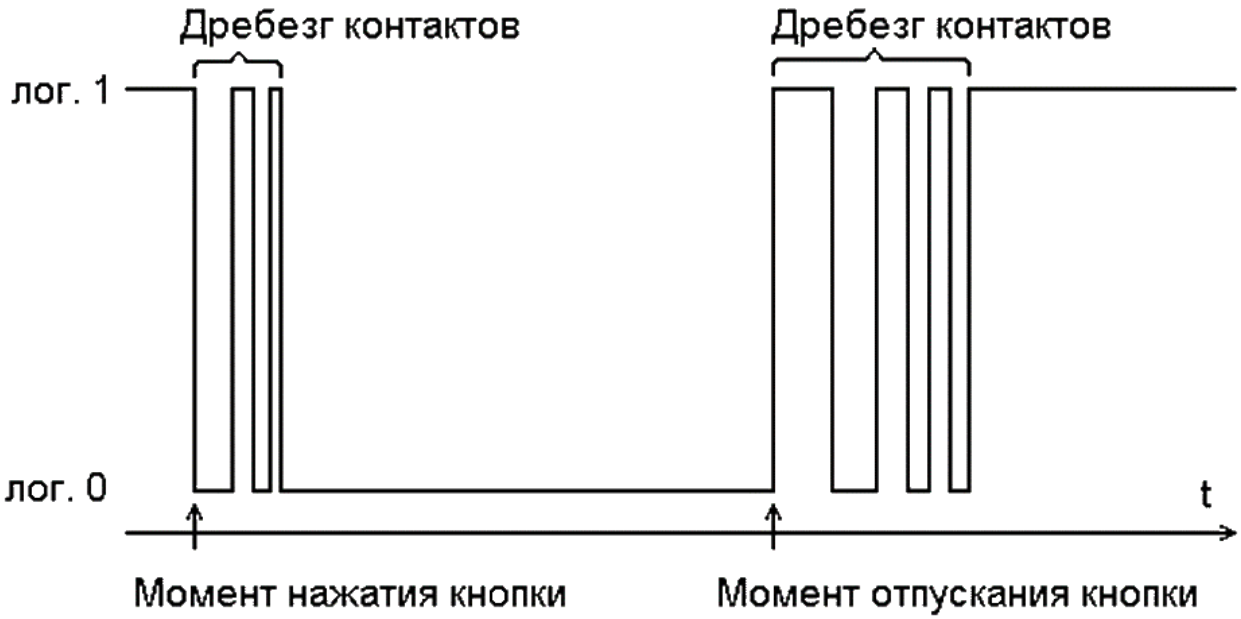

Рисунок 1.15. Диаграмма изменения сигнала при дребезге контактов

Возможная схема фильтра дребезга контактов представлена на рис. 1.16.

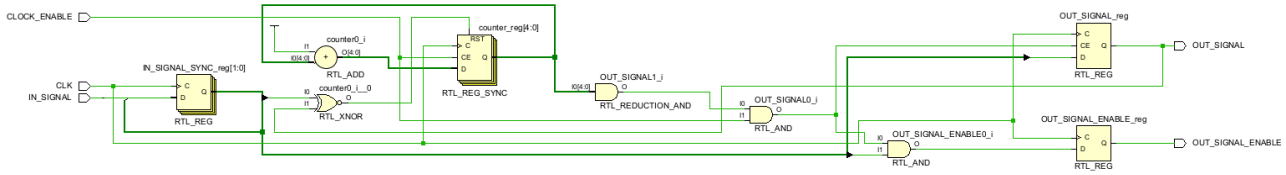

Рисунок 1.16. Схема фильтра дребезга контактов

Асинхронный сигнал проходит через блок синхронизации, описанный ранее. В схеме присутствуют два триггера, значения с которых поступают на выход. Один триггер служит для хранения сигнала, который был введен с кнопки, а второй формирует однотактный импульс, который служит индикатором валидности данных на другом триггере (речь идёт об уровне логической единицы). Значение на триггер «OUT_SIGNAL_reg» перезаписывается с выхода синхронизатора в том случае, если значение на выходе синхронизатора поменялось и продержалось на нём достаточно долго. За подсчёт времени поддержания стабильного уровня сигнала на выходе синхронизатора отвечает счётчик (на схеме представлен в виде регистра RTL_REG_SYNC в связке с сумматором, который выполняет роль инкрементатора). Разрядность счётчика в такой схеме определяет время

контроля стабильного сигнала на выходе синхронизатора. Счётчик сбрасывается всякий раз, когда значение на выходе синхронизатора совпадает со значением на выходе триггера «OUT_SIGNAL_reg». Вход «CLOCK_ENABLE» отвечает за контроль частоты работы устройства.

1.1.3. Движковые переключатели

Ползунковый (движковый) переключатель — это механический переключатель, который перемещается из открытого (выключенного) положения в закрытое (включенное) положение и позволяет контролировать ток в цепи без необходимости вручную сращивать или резать провода.

Ползунковый переключатель — это переключатель с постоянным контактом, в котором переключатель остается в одном состоянии до тех пор, пока он не будет приведен в действие в новом состоянии, где он остается до тех пор, пока не будет приведен в действие снова. Его размеры бывают сверхминиатюрными, миниатюрными и стандартными.

Ползунковый переключатель лучше всего использовать для управления текущим потоком в небольших проектах. Существует множество типов клемм для ползунковых переключателей, в том числе проходные, проволочные выводы, клеммы под пайку, винтовые клеммы, быстроразъемные или ножевые клеммы, технология поверхностного монтажа и переключатели для панельного монтажа. Рукоятка переключателя либо поднята, либо утоплена, в зависимости от типа привода. Выбор скрытого или приподнятого переключателя зависит от предполагаемого применения переключателя. Функции ползункового переключателя включают в себя контрольные лампы, переключатели с подсветкой, протирочные контакты и временные задержки.

На отладочной плате Xilinx/AMD Nexys A7 присутствуют шестнадцать ползунковых переключателей. Схема подключения представлена на рис. 1.17.

Принцип подключения к модулю на языке Verilog прост: необходимо указать соответствующие порты или порт в качестве входных, а далее снимать с них значение. Если использовать движковые переключатели в качестве управляющих механизмов, то стоит позаботиться о дребезге контактов. Рекомендуется для данной лабораторной работы использовать переключатели для ввода значений следующим образом: вначале выставить нужное значение на переключателях, а затем подтвердить введенное значение при помощи кнопки. Наличие фильтра дребезга контактов на кнопке позволит не отслеживать подобные ситуации на переключателях при таком подходе.

Стоит также отметить, что два из шестнадцати ползунковых переключателей соединены с напряжением питания 1.8 В, в то время как

остальные соединены с напряжением питания 3.3 В. Речь идёт о переключателях под номерами восемь и девять. Стоит учитывать этот факт при написании файла проектных ограничений.

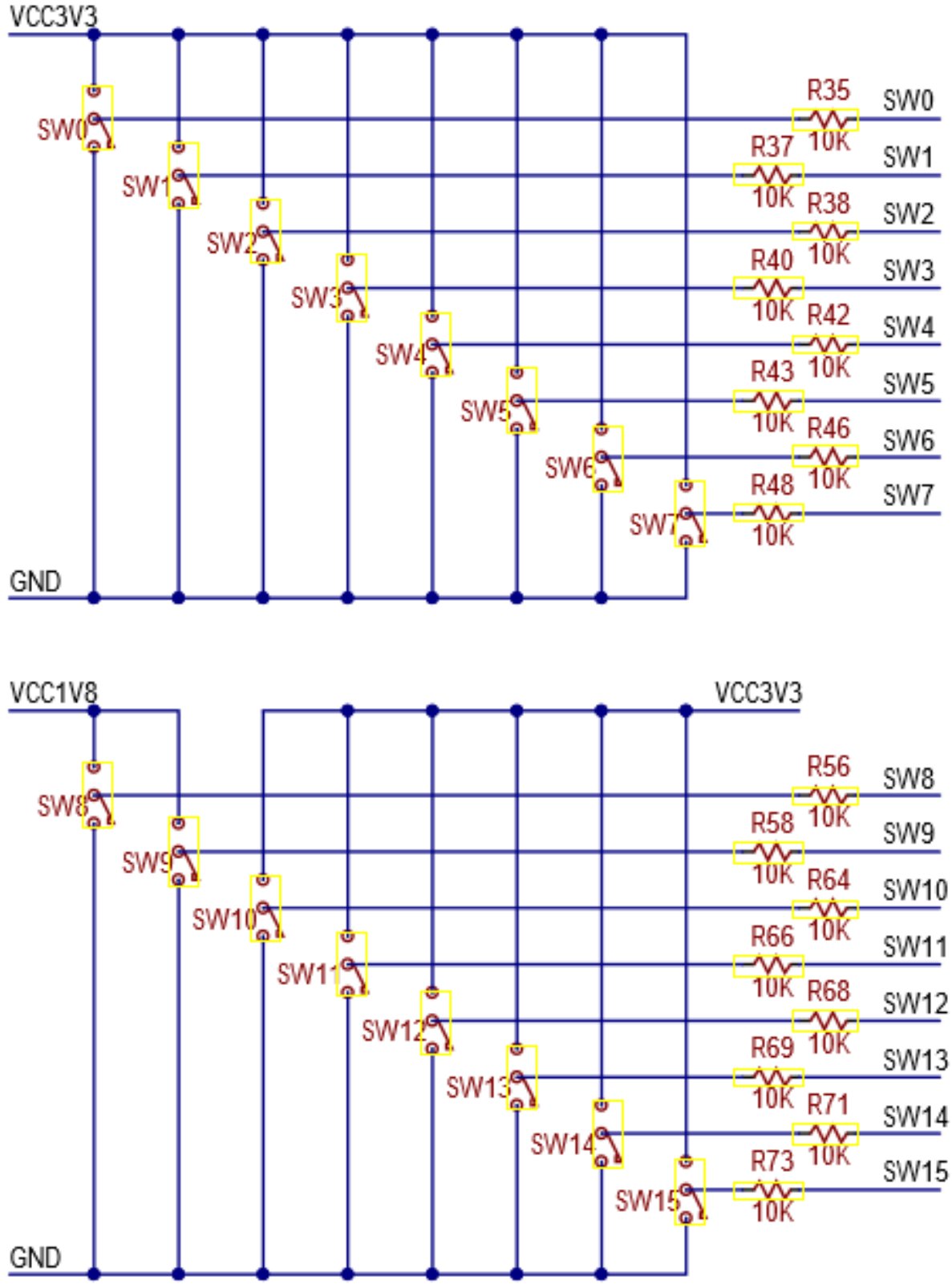

Рисунок 1.17. Схема подключения движковых переключателей на отладочной плате Xilinx/AMD Nexys A7

1.1.4. Файл проектных ограничений

Файлы проектных ограничений используются для того, чтобы описать некоторые свойства для проекта, которые не могут быть заданы при помощи языков описания аппаратуры. В САПР Vivado [1, 3] в качестве файлов проектных ограничений используются файлы с расширением «.xdc». представляют собой комбинацию стандартных отраслевых ограничений дизайна Synopsys (SDC) и собственные физические ограничения Xilinx/AMD [1, 3]. Формат xdc является расширением языка tcl (Tool Command Language), который широко используется в современных САПР.

В простейшем случае файлы проектных ограничений могут быть использованы для связи входов / выходов схемы с выходами ПЛИС, а также для установки электрического стандарта для конкретного порта. С этой целью используются две команды:

1) «set_property» — установка параметра (или параметров) для конкретной ресурса (ports, cells, pins, nets);
2) «get_ports» — создаёт список внешних выводов.

Для установки соединения используется параметр «PACKAGE_PIN». Для установки электрического стандарта используется параметр «IOSTANDARD». Например, в листинге 1.1 показан набор команд для установки связи между портом «sw[6]» некоторого модуля и пином «V6» ПЛИС; электрический стандарт устанавливается для напряжения высокого уровня 3.3 В.

Листинг 1.1. Пример команд для файла проектных ограничений

```
set_property PACKAGE_PIN V6 [get_ports {sw[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
```

Помимо соединений требуется создать тактовый генератор. Для этой цели используется команда «create_clock». Синтаксис команды приведён в листинге 1.2.

Листинг 1.2. Синтаксис команды «create_clock»

```
create_clock -period <период синхросигнала> [-name <имя порта синхросигнала>]
[-waveform <время восходящего фронта> <время нисходящего фронта> …]
[-add] [-quiet] [-verbose][objects]
```

Параметр «add» является опцией добавления нового синхросигнала. Параметр «quiet» отвечает за тихий режим исполнения команды. Параметр «verbose» отвечает за временное снятие ограничений по выводу сообщений и возврат всех сообщений после вызова данной команды. Параметр «objects» отвечает за порты, контакты или соединения, связанные с данным синхросигналом.

Пример создания тактового генератора представлен в листинге 1.3.

Листинг 1.3. Пример создания тактового генератора

```
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
CLK]
```

Набор проектных ограничений, которые в основном используются при установке связей между портами модуля верхнего уровня и пинами ПЛИС на отладочной плате, предоставляется производителем в виде отдельного «мастер-файла» с форматом xdc. Ниже (листинг 1.4) приведён исходный код «мастер-файла» для отладочной платы Xilinx/AMD Nexys A7. В нём представлены основные команды для соединения всех пинов ПЛИС Artix-7 xc7a100tcsg324-1I.

Листинг 1.4. «Мастер-файл» для отладочной платы Xilinx/AMD Nexys A7

```
## Clock signal
#set_property -dict { PACKAGE_PIN E3     IOSTANDARD LVCMOS33 } [get_ports {
CLK100MHZ }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
{CLK100MHZ}];


## Switches
#set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0]
}]; #IO_L24N_T3_RS0_15 Sch=sw[0]
#set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1]
}]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
#set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2]
}]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
#set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3]
}]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
#set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4]
}]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5]
}]; #IO_L7N_T1_D10_14 Sch=sw[5]
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6]
}]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
#set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7]
}]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8]
}]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9]
}]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16    IOSTANDARD LVCMOS33 } [get_ports {
SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13    IOSTANDARD LVCMOS33 } [get_ports {
SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6     IOSTANDARD LVCMOS33 } [get_ports {
SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12    IOSTANDARD LVCMOS33 } [get_ports {
SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11    IOSTANDARD LVCMOS33 } [get_ports {
SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
#set_property -dict { PACKAGE_PIN V10    IOSTANDARD LVCMOS33 } [get_ports {
SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]
```

Продолжение листинга 1.4

```
## LEDs
#set_property -dict { PACKAGE_PIN H17    IOSTANDARD LVCMOS33 } [get_ports {
LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
#set_property -dict { PACKAGE_PIN K15    IOSTANDARD LVCMOS33 } [get_ports {
LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
#set_property -dict { PACKAGE_PIN J13    IOSTANDARD LVCMOS33 } [get_ports {
LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
#set_property -dict { PACKAGE_PIN N14    IOSTANDARD LVCMOS33 } [get_ports {
LED[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
#set_property -dict { PACKAGE_PIN R18    IOSTANDARD LVCMOS33 } [get_ports {
LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
#set_property -dict { PACKAGE_PIN V17    IOSTANDARD LVCMOS33 } [get_ports {
LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
#set_property -dict { PACKAGE_PIN U17    IOSTANDARD LVCMOS33 } [get_ports {
LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
#set_property -dict { PACKAGE_PIN U16    IOSTANDARD LVCMOS33 } [get_ports {
LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
#set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVCMOS33 } [get_ports {
LED[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
#set_property -dict { PACKAGE_PIN T15    IOSTANDARD LVCMOS33 } [get_ports {
LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
#set_property -dict { PACKAGE_PIN U14    IOSTANDARD LVCMOS33 } [get_ports {
LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
#set_property -dict { PACKAGE_PIN T16    IOSTANDARD LVCMOS33 } [get_ports {
LED[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
#set_property -dict { PACKAGE_PIN V15    IOSTANDARD LVCMOS33 } [get_ports {
LED[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
#set_property -dict { PACKAGE_PIN V14    IOSTANDARD LVCMOS33 } [get_ports {
LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
#set_property -dict { PACKAGE_PIN V12    IOSTANDARD LVCMOS33 } [get_ports {
LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
#set_property -dict { PACKAGE_PIN V11    IOSTANDARD LVCMOS33 } [get_ports {
LED[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]


## RGB LEDs
#set_property -dict { PACKAGE_PIN R12    IOSTANDARD LVCMOS33 } [get_ports {
LED16_B }]; #IO_L5P_T0_D06_14 Sch=led16_b
#set_property -dict { PACKAGE_PIN M16    IOSTANDARD LVCMOS33 } [get_ports {
LED16_G }]; #IO_L10P_T1_D14_14 Sch=led16_g
#set_property -dict { PACKAGE_PIN N15    IOSTANDARD LVCMOS33 } [get_ports {
LED16_R }]; #IO_L11P_T1_SRCC_14 Sch=led16_r
#set_property -dict { PACKAGE_PIN G14    IOSTANDARD LVCMOS33 } [get_ports {
LED17_B }]; #IO_L15N_T2_DQS_ADV_B_15 Sch=led17_b
#set_property -dict { PACKAGE_PIN R11    IOSTANDARD LVCMOS33 } [get_ports {
LED17_G }]; #IO_0_14 Sch=led17_g
#set_property -dict { PACKAGE_PIN N16    IOSTANDARD LVCMOS33 } [get_ports {
LED17_R }]; #IO_L11N_T1_SRCC_14 Sch=led17_r


## 7 segment display
#set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { CA
}]; #IO_L24N_T3_A00_D16_14 Sch=ca
```

Продолжение листинга 1.4

```
#set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { CB
}]; #IO_25_14 Sch=cb
#set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { CC
}]; #IO_25_15 Sch=cc
#set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { CD
}]; #IO_L17P_T2_A26_15 Sch=cd
#set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { CE
}]; #IO_L13P_T2_MRCC_14 Sch=ce
#set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { CF
}]; #IO_L19P_T3_A10_D26_14 Sch=cf
#set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { CG
}]; #IO_L4P_T0_D04_14 Sch=cg
#set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { DP
}]; #IO_L19N_T3_A21_VREF_15 Sch=dp
#set_property -dict { PACKAGE_PIN J17  IOSTANDARD LVCMOS33 } [get_ports { AN[0]
}]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
#set_property -dict { PACKAGE_PIN J18  IOSTANDARD LVCMOS33 } [get_ports { AN[1]
}]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
#set_property -dict { PACKAGE_PIN T9   IOSTANDARD LVCMOS33 } [get_ports { AN[2]
}]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
#set_property -dict { PACKAGE_PIN J14  IOSTANDARD LVCMOS33 } [get_ports { AN[3]
}]; #IO_L19P_T3_A22_15 Sch=an[3]
#set_property -dict { PACKAGE_PIN P14  IOSTANDARD LVCMOS33 } [get_ports { AN[4]
}]; #IO_L8N_T1_D12_14 Sch=an[4]
#set_property -dict { PACKAGE_PIN T14  IOSTANDARD LVCMOS33 } [get_ports { AN[5]
}]; #IO_L14P_T2_SRCC_14 Sch=an[5]
#set_property -dict { PACKAGE_PIN K2   IOSTANDARD LVCMOS33 } [get_ports { AN[6]
}]; #IO_L23P_T3_35 Sch=an[6]
#set_property -dict { PACKAGE_PIN U13  IOSTANDARD LVCMOS33 } [get_ports { AN[7]
}]; #IO_L23N_T3_A02_D18_14 Sch=an[7]


## Buttons
#set_property -dict { PACKAGE_PIN C12    IOSTANDARD LVCMOS33 } [get_ports {
CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn
#set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTNC
}]; #IO_L9P_T1_DQS_14 Sch=btnc
#set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { BTNU
}]; #IO_L4N_T0_D05_14 Sch=btnu
#set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { BTNL
}]; #IO_L12P_T1_MRCC_14 Sch=btnl
#set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { BTNR
}]; #IO_L10N_T1_D15_14 Sch=btnr
#set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports { BTND
}]; #IO_L9N_T1_DQS_D13_14 Sch=btnd



## Pmod Headers
## Pmod Header JA
#set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33 } [get_ports { JA[1]
}]; #IO_L20N_T3_A19_15 Sch=ja[1]
```

Продолжение листинга 1.4

```
#set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { JA[2] }]; #IO_L21N_T3_DQS_A18_15 Sch=ja[2]
#set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { JA[3] }]; #IO_L21P_T3_DQS_15 Sch=ja[3]
#set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { JA[4] }]; #IO_L18N_T2_A23_15 Sch=ja[4]
#set_property -dict { PACKAGE_PIN D17   IOSTANDARD LVCMOS33 } [get_ports { JA[7] }]; #IO_L16N_T2_A27_15 Sch=ja[7]
#set_property -dict { PACKAGE_PIN E17   IOSTANDARD LVCMOS33 } [get_ports { JA[8] }]; #IO_L16P_T2_A28_15 Sch=ja[8]
#set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { JA[9] }]; #IO_L22N_T3_A16_15 Sch=ja[9]
#set_property -dict { PACKAGE_PIN G18    IOSTANDARD LVCMOS33 } [get_ports { JA[10] }]; #IO_L22P_T3_A17_15 Sch=ja[10]


## Pmod Header JB
#set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { JB[1] }]; #IO_L1P_T0_AD0P_15 Sch=jb[1]
#set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { JB[2] }]; #IO_L14N_T2_SRCC_15 Sch=jb[2]
#set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { JB[3] }]; #IO_L13N_T2_MRCC_15 Sch=jb[3]
#set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { JB[4] }]; #IO_L15P_T2_DQS_15 Sch=jb[4]
#set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports { JB[7] }]; #IO_L11N_T1_SRCC_15 Sch=jb[7]
#set_property -dict { PACKAGE_PIN F13   IOSTANDARD LVCMOS33 } [get_ports { JB[8] }]; #IO_L5P_T0_AD9P_15 Sch=jb[8]
#set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { JB[9] }]; #IO_0_15 Sch=jb[9]
#set_property -dict { PACKAGE_PIN H16    IOSTANDARD LVCMOS33 } [get_ports { JB[10] }]; #IO_L13P_T2_MRCC_15 Sch=jb[10]


## Pmod Header JC
#set_property -dict { PACKAGE_PIN K1    IOSTANDARD LVCMOS33 } [get_ports { JC[1] }]; #IO_L23N_T3_35 Sch=jc[1]
#set_property -dict { PACKAGE_PIN F6    IOSTANDARD LVCMOS33 } [get_ports { JC[2] }]; #IO_L19N_T3_VREF_35 Sch=jc[2]
#set_property -dict { PACKAGE_PIN J2    IOSTANDARD LVCMOS33 } [get_ports { JC[3] }]; #IO_L22N_T3_35 Sch=jc[3]
#set_property -dict { PACKAGE_PIN G6    IOSTANDARD LVCMOS33 } [get_ports { JC[4] }]; #IO_L19P_T3_35 Sch=jc[4]
#set_property -dict { PACKAGE_PIN E7    IOSTANDARD LVCMOS33 } [get_ports { JC[7] }]; #IO_L6P_T0_35 Sch=jc[7]
#set_property -dict { PACKAGE_PIN J3    IOSTANDARD LVCMOS33 } [get_ports { JC[8] }]; #IO_L22P_T3_35 Sch=jc[8]
#set_property -dict { PACKAGE_PIN J4    IOSTANDARD LVCMOS33 } [get_ports { JC[9] }]; #IO_L21P_T3_DQS_35 Sch=jc[9]
#set_property -dict { PACKAGE_PIN E6     IOSTANDARD LVCMOS33 } [get_ports { JC[10] }]; #IO_L5P_T0_AD13P_35 Sch=jc[10]
```

Продолжение листинга 1.4

```
## Pmod Header JD
#set_property -dict { PACKAGE_PIN H4    IOSTANDARD LVCMOS33 } [get_ports { JD[1] }]; #IO_L21N_T3_DQS_35 Sch=jd[1]
#set_property -dict { PACKAGE_PIN H1    IOSTANDARD LVCMOS33 } [get_ports { JD[2] }]; #IO_L17P_T2_35 Sch=jd[2]
#set_property -dict { PACKAGE_PIN G1    IOSTANDARD LVCMOS33 } [get_ports { JD[3] }]; #IO_L17N_T2_35 Sch=jd[3]
#set_property -dict { PACKAGE_PIN G3    IOSTANDARD LVCMOS33 } [get_ports { JD[4] }]; #IO_L20N_T3_35 Sch=jd[4]
#set_property -dict { PACKAGE_PIN H2    IOSTANDARD LVCMOS33 } [get_ports { JD[7] }]; #IO_L15P_T2_DQS_35 Sch=jd[7]
#set_property -dict { PACKAGE_PIN G4    IOSTANDARD LVCMOS33 } [get_ports { JD[8] }]; #IO_L20P_T3_35 Sch=jd[8]
#set_property -dict { PACKAGE_PIN G2    IOSTANDARD LVCMOS33 } [get_ports { JD[9] }]; #IO_L15N_T2_DQS_35 Sch=jd[9]
#set_property -dict { PACKAGE_PIN F3     IOSTANDARD LVCMOS33 } [get_ports { JD[10] }]; #IO_L13N_T2_MRCC_35 Sch=jd[10]
## Pmod Header JXADC
#set_property -dict { PACKAGE_PIN A14    IOSTANDARD LVCMOS33 } [get_ports { XA_N[1] }]; #IO_L9N_T1_DQS_AD3N_15 Sch=xa_n[1]
#set_property -dict { PACKAGE_PIN A13    IOSTANDARD LVCMOS33 } [get_ports { XA_P[1] }]; #IO_L9P_T1_DQS_AD3P_15 Sch=xa_p[1]
#set_property -dict { PACKAGE_PIN A16    IOSTANDARD LVCMOS33 } [get_ports { XA_N[2] }]; #IO_L8N_T1_AD10N_15 Sch=xa_n[2]
#set_property -dict { PACKAGE_PIN A15    IOSTANDARD LVCMOS33 } [get_ports { XA_P[2] }]; #IO_L8P_T1_AD10P_15 Sch=xa_p[2]
#set_property -dict { PACKAGE_PIN B17    IOSTANDARD LVCMOS33 } [get_ports { XA_N[3] }]; #IO_L7N_T1_AD2N_15 Sch=xa_n[3]
#set_property -dict { PACKAGE_PIN B16    IOSTANDARD LVCMOS33 } [get_ports { XA_P[3] }]; #IO_L7P_T1_AD2P_15 Sch=xa_p[3]
#set_property -dict { PACKAGE_PIN A18    IOSTANDARD LVCMOS33 } [get_ports { XA_N[4] }]; #IO_L10N_T1_AD11N_15 Sch=xa_n[4]
#set_property -dict { PACKAGE_PIN B18    IOSTANDARD LVCMOS33 } [get_ports { XA_P[4] }]; #IO_L10P_T1_AD11P_15 Sch=xa_p[4]
## VGA Connector
#set_property -dict { PACKAGE_PIN A3     IOSTANDARD LVCMOS33 } [get_ports { VGA_R[0] }]; #IO_L8N_T1_AD14N_35 Sch=vga_r[0]
#set_property -dict { PACKAGE_PIN B4     IOSTANDARD LVCMOS33 } [get_ports { VGA_R[1] }]; #IO_L7N_T1_AD6N_35 Sch=vga_r[1]
#set_property -dict { PACKAGE_PIN C5     IOSTANDARD LVCMOS33 } [get_ports { VGA_R[2] }]; #IO_L1N_T0_AD4N_35 Sch=vga_r[2]
#set_property -dict { PACKAGE_PIN A4     IOSTANDARD LVCMOS33 } [get_ports { VGA_R[3] }]; #IO_L8P_T1_AD14P_35 Sch=vga_r[3]
#set_property -dict { PACKAGE_PIN C6     IOSTANDARD LVCMOS33 } [get_ports { VGA_G[0] }]; #IO_L1P_T0_AD4P_35 Sch=vga_g[0]
#set_property -dict { PACKAGE_PIN A5     IOSTANDARD LVCMOS33 } [get_ports { VGA_G[1] }]; #IO_L3N_T0_DQS_AD5N_35 Sch=vga_g[1]
#set_property -dict { PACKAGE_PIN B6     IOSTANDARD LVCMOS33 } [get_ports { VGA_G[2] }]; #IO_L2N_T0_AD12N_35 Sch=vga_g[2]
#set_property -dict { PACKAGE_PIN A6     IOSTANDARD LVCMOS33 } [get_ports { VGA_G[3] }]; #IO_L3P_T0_DQS_AD5P_35 Sch=vga_g[3]
```

Продолжение листинга 1.4

```
#set_property -dict { PACKAGE_PIN B7     IOSTANDARD LVCMOS33 } [get_ports {
VGA_B[0] }]; #IO_L2P_T0_AD12P_35 Sch=vga_b[0]
#set_property -dict { PACKAGE_PIN C7     IOSTANDARD LVCMOS33 } [get_ports {
VGA_B[1] }]; #IO_L4N_T0_35 Sch=vga_b[1]
#set_property -dict { PACKAGE_PIN D7     IOSTANDARD LVCMOS33 } [get_ports {
VGA_B[2] }]; #IO_L6N_T0_VREF_35 Sch=vga_b[2]
#set_property -dict { PACKAGE_PIN D8     IOSTANDARD LVCMOS33 } [get_ports {
VGA_B[3] }]; #IO_L4P_T0_35 Sch=vga_b[3]
#set_property -dict { PACKAGE_PIN B11    IOSTANDARD LVCMOS33 } [get_ports {
VGA_HS }]; #IO_L4P_T0_15 Sch=vga_hs
#set_property -dict { PACKAGE_PIN B12    IOSTANDARD LVCMOS33 } [get_ports {
VGA_VS }]; #IO_L3N_T0_DQS_AD1N_15 Sch=vga_vs


## Micro SD Connector
#set_property -dict { PACKAGE_PIN E2     IOSTANDARD LVCMOS33 } [get_ports {
SD_RESET }]; #IO_L14P_T2_SRCC_35 Sch=sd_reset
#set_property -dict { PACKAGE_PIN A1    IOSTANDARD LVCMOS33 } [get_ports { SD_CD
}]; #IO_L9N_T1_DQS_AD7N_35 Sch=sd_cd
#set_property -dict { PACKAGE_PIN B1     IOSTANDARD LVCMOS33 } [get_ports {
SD_SCK }]; #IO_L9P_T1_DQS_AD7P_35 Sch=sd_sck
#set_property -dict { PACKAGE_PIN C1     IOSTANDARD LVCMOS33 } [get_ports {
SD_CMD }]; #IO_L16N_T2_35 Sch=sd_cmd
#set_property -dict { PACKAGE_PIN C2     IOSTANDARD LVCMOS33 } [get_ports {
SD_DAT[0] }]; #IO_L16P_T2_35 Sch=sd_dat[0]
#set_property -dict { PACKAGE_PIN E1     IOSTANDARD LVCMOS33 } [get_ports {
SD_DAT[1] }]; #IO_L18N_T2_35 Sch=sd_dat[1]
#set_property -dict { PACKAGE_PIN F1     IOSTANDARD LVCMOS33 } [get_ports {
SD_DAT[2] }]; #IO_L18P_T2_35 Sch=sd_dat[2]
#set_property -dict { PACKAGE_PIN D2     IOSTANDARD LVCMOS33 } [get_ports {
SD_DAT[3] }]; #IO_L14N_T2_SRCC_35 Sch=sd_dat[3]


## Accelerometer
#set_property -dict { PACKAGE_PIN E15    IOSTANDARD LVCMOS33 } [get_ports {
ACL_MISO }]; #IO_L11P_T1_SRCC_15 Sch=acl_miso
#set_property -dict { PACKAGE_PIN F14    IOSTANDARD LVCMOS33 } [get_ports {
ACL_MOSI }]; #IO_L5N_T0_AD9N_15 Sch=acl_mosi
#set_property -dict { PACKAGE_PIN F15    IOSTANDARD LVCMOS33 } [get_ports {
ACL_SCLK }]; #IO_L14P_T2_SRCC_15 Sch=acl_sclk
#set_property -dict { PACKAGE_PIN D15    IOSTANDARD LVCMOS33 } [get_ports {
ACL_CSN }]; #IO_L12P_T1_MRCC_15 Sch=acl_csn
#set_property -dict { PACKAGE_PIN B13    IOSTANDARD LVCMOS33 } [get_ports {
ACL_INT[1] }]; #IO_L2P_T0_AD8P_15 Sch=acl_int[1]
#set_property -dict { PACKAGE_PIN C16    IOSTANDARD LVCMOS33 } [get_ports {
ACL_INT[2] }]; #IO_L20P_T3_A20_15 Sch=acl_int[2]


## Temperature Sensor
#set_property -dict { PACKAGE_PIN C14    IOSTANDARD LVCMOS33 } [get_ports {
TMP_SCL }]; #IO_L1N_T0_AD0N_15 Sch=tmp_scl
#set_property -dict { PACKAGE_PIN C15    IOSTANDARD LVCMOS33 } [get_ports {
TMP_SDA }]; #IO_L12N_T1_MRCC_15 Sch=tmp_sda
```

Продолжение листинга 1.4

```
#set_property -dict { PACKAGE_PIN D13    IOSTANDARD LVCMOS33 } [get_ports {
TMP_INT }]; #IO_L6N_T0_VREF_15 Sch=tmp_int
#set_property -dict { PACKAGE_PIN B14    IOSTANDARD LVCMOS33 } [get_ports {
TMP_CT }]; #IO_L2N_T0_AD8N_15 Sch=tmp_ct


## Omnidirectional Microphone
#set_property -dict { PACKAGE_PIN J5    IOSTANDARD LVCMOS33 } [get_ports { M_CLK
}]; #IO_25_35 Sch=m_clk
#set_property -dict { PACKAGE_PIN H5     IOSTANDARD LVCMOS33 } [get_ports {
M_DATA }]; #IO_L24N_T3_35 Sch=m_data
#set_property -dict { PACKAGE_PIN F5     IOSTANDARD LVCMOS33 } [get_ports {
M_LRSEL }]; #IO_0_35 Sch=m_lrsel


## PWM Audio Amplifier
#set_property -dict { PACKAGE_PIN A11    IOSTANDARD LVCMOS33 } [get_ports {
AUD_PWM }]; #IO_L4N_T0_15 Sch=aud_pwm
#set_property -dict { PACKAGE_PIN D12    IOSTANDARD LVCMOS33 } [get_ports {
AUD_SD }]; #IO_L6P_T0_15 Sch=aud_sd


## USB-RS232 Interface
#set_property -dict { PACKAGE_PIN C4     IOSTANDARD LVCMOS33 } [get_ports {
UART_TXD_IN }]; #IO_L7P_T1_AD6P_35 Sch=uart_txd_in
#set_property -dict { PACKAGE_PIN D4     IOSTANDARD LVCMOS33 } [get_ports {
UART_RXD_OUT }]; #IO_L11N_T1_SRCC_35 Sch=uart_rxd_out
#set_property -dict { PACKAGE_PIN D3     IOSTANDARD LVCMOS33 } [get_ports {
UART_CTS }]; #IO_L12N_T1_MRCC_35 Sch=uart_cts
#set_property -dict { PACKAGE_PIN E5     IOSTANDARD LVCMOS33 } [get_ports {
UART_RTS }]; #IO_L5N_T0_AD13N_35 Sch=uart_rts


## USB HID (PS/2)
#set_property -dict { PACKAGE_PIN F4     IOSTANDARD LVCMOS33 } [get_ports {
PS2_CLK }]; #IO_L13P_T2_MRCC_35 Sch=ps2_clk
#set_property -dict { PACKAGE_PIN B2     IOSTANDARD LVCMOS33 } [get_ports {
PS2_DATA }]; #IO_L10N_T1_AD15N_35 Sch=ps2_data


## SMSC Ethernet PHY
#set_property -dict { PACKAGE_PIN C9     IOSTANDARD LVCMOS33 } [get_ports {
ETH_MDC }]; #IO_L11P_T1_SRCC_16 Sch=eth_mdc
#set_property -dict { PACKAGE_PIN A9     IOSTANDARD LVCMOS33 } [get_ports {
ETH_MDIO }]; #IO_L14N_T2_SRCC_16 Sch=eth_mdio
#set_property -dict { PACKAGE_PIN B3     IOSTANDARD LVCMOS33 } [get_ports {
ETH_RSTN }]; #IO_L10P_T1_AD15P_35 Sch=eth_rstn
#set_property -dict { PACKAGE_PIN D9     IOSTANDARD LVCMOS33 } [get_ports {
ETH_CRSDV }]; #IO_L6N_T0_VREF_16 Sch=eth_crsdv
#set_property -dict { PACKAGE_PIN C10    IOSTANDARD LVCMOS33 } [get_ports {
ETH_RXERR }]; #IO_L13N_T2_MRCC_16 Sch=eth_rxerr
#set_property -dict { PACKAGE_PIN C11    IOSTANDARD LVCMOS33 } [get_ports {
ETH_RXD[0] }]; #IO_L13P_T2_MRCC_16 Sch=eth_rxd[0]
#set_property -dict { PACKAGE_PIN D10    IOSTANDARD LVCMOS33 } [get_ports {
ETH_RXD[1] }]; #IO_L19N_T3_VREF_16 Sch=eth_rxd[1]
```

Продолжение листинга 1.4

```
#set_property -dict { PACKAGE_PIN B9     IOSTANDARD LVCMOS33 } [get_ports {
ETH_TXEN }]; #IO_L11N_T1_SRCC_16 Sch=eth_txen
#set_property -dict { PACKAGE_PIN A10    IOSTANDARD LVCMOS33 } [get_ports {
ETH_TXD[0] }]; #IO_L14P_T2_SRCC_16 Sch=eth_txd[0]
#set_property -dict { PACKAGE_PIN A8     IOSTANDARD LVCMOS33 } [get_ports {
ETH_TXD[1] }]; #IO_L12N_T1_MRCC_16 Sch=eth_txd[1]
#set_property -dict { PACKAGE_PIN D5     IOSTANDARD LVCMOS33 } [get_ports {
ETH_REFCLK }]; #IO_L11P_T1_SRCC_35 Sch=eth_refclk
#set_property -dict { PACKAGE_PIN B8     IOSTANDARD LVCMOS33 } [get_ports {
ETH_INTN }]; #IO_L12P_T1_MRCC_16 Sch=eth_intn


## Quad SPI Flash
#set_property -dict { PACKAGE_PIN K17    IOSTANDARD LVCMOS33 } [get_ports {
QSPI_DQ[0] }]; #IO_L1P_T0_D00_MOSI_14 Sch=qspi_dq[0]
#set_property -dict { PACKAGE_PIN K18    IOSTANDARD LVCMOS33 } [get_ports {
QSPI_DQ[1] }]; #IO_L1N_T0_D01_DIN_14 Sch=qspi_dq[1]
#set_property -dict { PACKAGE_PIN L14    IOSTANDARD LVCMOS33 } [get_ports {
QSPI_DQ[2] }]; #IO_L2P_T0_D02_14 Sch=qspi_dq[2]
#set_property -dict { PACKAGE_PIN M14    IOSTANDARD LVCMOS33 } [get_ports {
QSPI_DQ[3] }]; #IO_L2N_T0_D03_14 Sch=qspi_dq[3]
#set_property -dict { PACKAGE_PIN L13    IOSTANDARD LVCMOS33 } [get_ports {
QSPI_CSN }]; #IO_L6P_T0_FCS_B_14 Sch=qspi_csn
```

1.1.5. Алгоритм генерации прошивки ПЛИС формата «.bit» для САПР Vivado

Для генерации файла прошивки необходимо:

1. Провести синтез проекта, выбрав опцию «Run Synthesis» в САПР Vivado.
2. Провести имплементацию проекта, выбрав опцию «Run Implementation» в САПР Vivado.
3. Запустить генерацию прошивки, выбрав опцию «Generate Bitstream» в САПР Vivado.

В случае успешного выполнения описанных процессов файл прошивки появится в папке с проектом и будет иметь следующее наименование:

<наименование модуля верхнего уровня>.bit

Также для удобства запуска выполнения всех процессов можно сразу перейти к шагу 3. САПР Vivado выведет предупреждение, что процессы синтеза и имплементации выполнялись давно или не выполнялись вовсе, для запуска исполнения этих процессов необходимо нажать на кнопку «Yes» в окне предупреждения.

1.2. Пример выполнения работы

1.2.1. Реализация

1.2.1.1 Модуль фильтра дребезга контактов

Данный модуль в качестве подмодуля будет включать в себя синхронизатор. Исходный код модуля фильтра дребезга контактов представлен в листинге 1.5.

Листинг 1.5. Модуль фильтра дребезга контактов

```
module FILTER #(size = 3) (
input CLK, CLOCK_ENABLE, IN_SIGNAL,
output reg OUT_SIGNAL, OUT_SIGNAL_ENABLE
);
reg [1:0] IN_SIGNAL_SYNC;
reg [size-1:0] counter;
initial
begin
    IN_SIGNAL_SYNC = 0; counter = 0;
    OUT_SIGNAL = 0; OUT_SIGNAL_ENABLE = 0;
end
always @(posedge CLK)
begin
    IN_SIGNAL_SYNC <= {IN_SIGNAL_SYNC[0], IN_SIGNAL};
    counter <= (IN_SIGNAL_SYNC[1] ~^ OUT_SIGNAL) ?
        {size{1'd0}} : (CLOCK_ENABLE ? counter + 1 : counter);
    if (&(counter) & CLOCK_ENABLE)
        OUT_SIGNAL <= IN_SIGNAL_SYNC[1];
    OUT_SIGNAL_ENABLE <= &(counter) & CLOCK_ENABLE & IN_SIGNAL_SYNC[1];
end
endmodule
```

В качестве параметра модулю передаётся размер счётчика, который определяет число тактов контроля стабильного сигнала на выходе синхронизатора. «IN_SIGNAL_SYNC» – двухбитный регистр, который выполняет роль синхронизатора. Регистр является сдвиговым, и значение на нём постепенно перемещается от младшего бита к старшему. Сигнал «OUT_SIGNAL_ENABLE» будет вставать в единицу в момент, когда: счётчик досчитает до своего максимума (запроверку этого факта отвечает побитовая конъюнкция выходов счётчика «counter»), а также сигнал на выходе синхронизатора должен быть равен логической единице при условии, что сигнал «CLOCK_ENABLE» разрешает просчёт схемы.

1.2.1.2 Модуль управления семисегментными индикаторами

Исходный код модуля управления семисегментными индикаторами представлен в листинге 1.6.

Листинг 1.6. Модуль управления семисегментными индикаторами

```
module SevenSegmentLED (
        input clk,
        input RESET,

        // Входное число
        input [31:0] NUMBER,

        // Маска отображения 7-сегментных индикаторов
        input [7:0] AN_MASK,

        output [7:0] AN,  // шина разрешающих входов анодов для всех
индикаторов
        output reg [6:0] SEG // шина катодов для одного индикатора
);

reg [7:0] AN_REG = 0;
assign AN = AN_REG | AN_MASK;

// Значение на счётчике обозначает номер разряда входного числа (номер
индикатора),
// который отображается (3-х разрядный, так как только 8 цифр можно
отобразить)
reg [2:0] digit_counter;

// Разветвитель входящего числа по группам из 4-х бит
wire [3:0] NUMBER_SPLITTER [0:7];// = {NUMBER[3:0], NUMBER[7:4], NUMBER[11:8],
NUMBER[15:12], NUMBER[19:16], NUMBER[23:20], NUMBER[27:24], NUMBER[31:28]};

genvar i;
generate
      for (i = 0; i < 8; i = i + 1)
      begin
            assign NUMBER_SPLITTER[i] = NUMBER[((i+1)*4-1)-:4];
    end
endgenerate

initial begin
      digit_counter = 0;
end

// Блок управления счётчиком индикаторов
always @(posedge clk)
      digit_counter <= RESET ? 0 : digit_counter + 3'b1;

/*
      В силу особенностей реализации на плате 7-сегментных индикаторов,
      все индикаторы имеют общий анод, который выбирает индикатор, а
      катоды соединены с общей шиной и подключены к каждому индикатору
*/
always @ (digit_counter)
begin
      /*
            Установка подсветки 7-сегментного индикатора
            в зависимости от дешифрованной цифры
      */
```

Продолжение листинга 1.6

```
      case (NUMBER_SPLITTER[digit_counter])
                  4'h0:  SEG <= 7'b1000000;
                  4'h1:  SEG <= 7'b1111001;
                  4'h2:  SEG <= 7'b0100100;
                  4'h3:  SEG <= 7'b0110000;
                  4'h4:  SEG <= 7'b0011001;
                  4'h5:  SEG <= 7'b0010010;
                  4'h6:  SEG <= 7'b0000010;
                  4'h7:  SEG <= 7'b1111000;
                  4'h8:  SEG <= 7'b0000000;
                  4'h9:  SEG <= 7'b0010000;
                  4'ha:  SEG <= 7'b0001000;
                  4'hb:  SEG <= 7'b0000011;
                  4'hc:  SEG <= 7'b1000110;
                  4'hd:  SEG <= 7'b0100001;
                  4'he:  SEG <= 7'b0000110;
                  4'hf:  SEG <= 7'b0001110;
               default:  SEG <= 7'b1111111;
      endcase

      /*
            Поскольку у всех 7-сегментных индикаторов один общий анод,
            то необходимо, в зависимости от номера разряда, который
            отображается в текущий дискретный момент времени, подать
            соответствующий сигнал на нужный индикатор.

            Чтобы осветить сегмент, анод должен находиться в высоком уровне, а
катод - в низком.
            Однако, поскольку используются транзисторы для подачи достаточного
тока
            в общую точку анода, вход разрешения для анода инвертируется.
      */
      //AN_REG <= ~(8'd1 << digit_counter);


      case (digit_counter)
                  3'd0: AN_REG <= 8'b11111110;
                  3'd1: AN_REG <= 8'b11111101;
                  3'd2: AN_REG <= 8'b11111011;
                  3'd3: AN_REG <= 8'b11110111;
                  3'd4: AN_REG <= 8'b11101111;
                  3'd5: AN_REG <= 8'b11011111;
                  3'd6: AN_REG <= 8'b10111111;
                  3'd7: AN_REG <= 8'b01111111;
               default: AN_REG <= 8'b11111111;
      endcase
end
endmodule
```

Для разделения входной шины используется оператор «generate». Управляющий сигнал для анодов вырабатывается при помощи дизъюнкции, поскольку действующим уровнем сигнала является логический ноль в силу конструктивных особенностей.

1.2.1.3 Модуль верхнего уровня

Исходный код модуля верхнего уровня представлен в листинге 1.7.

Листинг 1.7. Модуль верхнего уровня

```
module main(
      input clk,
      input btn_c,
      input [3:0] SW,
      output [7:0] AN,
      output [6:0] SEG
);

reg CLOCK_ENABLE = 0;
always @(posedge clk)
      CLOCK_ENABLE <= ~CLOCK_ENABLE;
wire btn_c_out, btn_c_out_enable;
FILTER #(
    .size(7)
) btn_c_filter
(
      .CLK(clk),
      .CLOCK_ENABLE(CLOCK_ENABLE),
      .IN_SIGNAL(btn_c),
      .OUT_SIGNAL(btn_c_out),
      .OUT_SIGNAL_ENABLE(btn_c_out_enable)
);

reg [31:0] shift_register;
reg [7:0] an_mask;
initial
begin
      shift_register = 0;
      an_mask <= 8'b11111111;
end
always@(posedge clk)
      if (btn_c_out_enable)
            begin
                  shift_register <= {shift_register[27:0], SW};
                  an_mask <= {an_mask[6:0], 1'b0};
            end

wire clk_div_out;
clk_div clk_div1 (
      .clk(clk),
      .clk_div(clk_div_out)
);

SevenSegmentLED seg(
      .clk(clk_div_out),
      .RESET(1'b0),
      .NUMBER(shift_register),
      .AN_MASK(an_mask),
      .AN(AN),
      .SEG(SEG)
);
endmodule
```

1.2.1.4 Файл проектных ограничений

Исходный код файла проектных ограничений представлен в листинге 1.8.

Листинг 1.8. Файл проектных ограничений

```
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
{clk}]

set_property IOSTANDARD LVCMOS33 [get_ports {clk}]
set_property PACKAGE_PIN E3 [get_ports {clk}]

set_property PACKAGE_PIN J15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[0]}]
set_property PACKAGE_PIN L16 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[1]}]
set_property PACKAGE_PIN M13 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[2]}]
set_property PACKAGE_PIN V10 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[3]}]

set_property PACKAGE_PIN N17 [get_ports {btn_c}]
set_property IOSTANDARD LVCMOS33 [get_ports {btn_c}]

set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN J17} [get_ports
{AN[0]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN J18} [get_ports
{AN[1]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN T9}  [get_ports
{AN[2]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN J14} [get_ports
{AN[3]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN P14} [get_ports
{AN[4]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN T14} [get_ports
{AN[5]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN K2}  [get_ports
{AN[6]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN U13} [get_ports
{AN[7]}]

set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN T10} [get_ports
{SEG[0]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN R10} [get_ports
{SEG[1]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN K16} [get_ports
{SEG[2]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN K13} [get_ports
{SEG[3]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN P15} [get_ports
{SEG[4]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN T11} [get_ports
{SEG[5]}]
set_property -dict {IOSTANDARD LVCMOS33        PACKAGE_PIN L18} [get_ports
{SEG[6]}]
```

1.3. Заключение

В результате выполнения лабораторной работы студентами был приобретён навык построения управляющего устройства для визуального отображения информации при помощи набора семисегментных индикаторов, изучены основные особенности считывания сигнала с физического устройства ввода — кнопки.

1.4. Список контрольных вопросов

1. Что такое светодиод?
2. Объясните принцип работы семисегментного индикатора.
3. Для чего необходимо синхронизировать асинхронный сигнал?
4. Объясните причину возникновения дребезга контактов.
5. Как фильтр дребезга контактов ждущего типа решает проблему дребезга контактов?
6. Какие этапы включает в себя процесс создания проекта с использованием ПЛИС?
7. Иные вопросы по теме работы.