**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**Федеральное государственное  
бюджетное образовательное учреждение высшего образования**

**«САНКТ-ПЕТЕРБУРГСКИЙ**

**ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ**

**им. проф. М. А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**О. Л. Неелова**

**«Проектирование элементов вычислительных систем на программируемых логических интегральных схемах (ПЛИС)»**

**ПРАКТИКУМ ПО ДИСЦИПЛИНЕ**

**«АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

Описание: Описание: C:\Users\XZ\Desktop\logo_1.wmf

**САНКТ-ПЕТЕРБУРГ**

**2022**

УДК 004.25 + 004.27 + 004.3 + 621.39 + 621.3.049.77

ББК 32.973.2

Рецензент:

кандидат технических наук, доцент кафедры радиосвязи и вещания

*Б. Г. Шамсиев*

*Утверждено редакционно-издательским советом СПбГУТ*

*в качестве учебного пособия*

**Неелова О. Л.**

|  |  |
| --- | --- |
|  | Практикум по дисциплине «Архитектура вычислительных систем» : / О. Л. Неелова; СПбГУТ. – СПб., 2022. – с.  Приводятся материалы для проведения лабораторно-практических занятий по дисциплинам «архитектура вычислительных систем» и «вычислительная техника», посвященных реализации модулей вычислительных систем в схемах программируемой логики.  Предназначено для подготовки бакалавров по на­правлению: 09.03.04 «Программная инженерия», 09.03.01 «Информатик и вычислительная техника».  **УДК 004.25 + 004.27 + 004.3 + 621.39 + 621.3.049.77**  **ББК 32.973.2** |

Неелова О. Л., 2022

© Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет

телекоммуникаций им. проф. М. А. Бонч-Бруевича», 2022

Содержание.

1. Вводное занятие стр. 4.
2. Руководство по работе с пакетом *ModelSim* стр. 8.
3. Лабораторная работа №1 стр. 13
4. Лабораторная работа №2 стр. 15
5. Лабораторная работа №3 стр. 19
6. Лабораторная работа №4 стр. 23
7. Лабораторная работа №5 стр. 26
8. Лабораторная работа №6 стр. 29
9. Приложение 1. Работа с пакетом *Quartus*15 стр. 30
10. П. 1. 1. Создание проекта стр. 30
11. П. 1. 2. Планировщик стр. 32
12. П. 1. 3. Программатор стр. 37
13. Приложение 2. Задания к л. р. № 6 стр. 39

Вводное занятие.

**Ознакомление с работой в пакете *ModelSim.***

Пакет *ModelSim* позволяет производить моделирование устройств по определенному заданию. Как правило, такие устройства представлены проектом из нескольких модулей, каждый из которых описывается отдельным файлом. Файлом верхнего уровня проекта является файл сборки, в который включаются все созданные модули. Каждый модуль проходит функциональную верификацию. Для этого на сохраненный в проекте и откомпилированный файл модуля пишется файл *testbench*. В этом файле на входы созданного модуля подаются тестовые сигналы с виртуального генератора тестового стенда, а к выходам подключается виртуальный монитор, на котором просматривается функциональная диаграмма устройства, описанного в модуле, рис. 1.

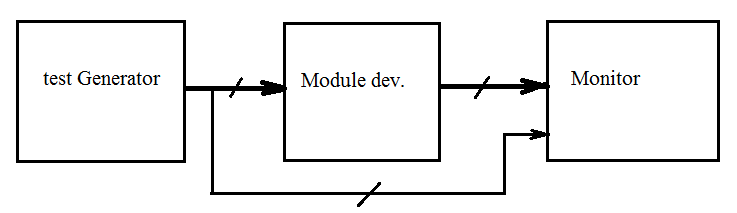


Рис. 1. Блок-схема тестового стенда.

Для примера проведем моделирование с проверкой функциональности схемы кодопреобразователя, позволяющего выводить формируемое на 4-разрядной шине двоичное число на семисегментный индикатор. Число при этом представляется в шестнадцатирисчном кодировании. При формировании кода следует учесть, что сегменты индикатора загораются при подаче на них уровня «0».Сегменты поименуем буквами латинского алфавита *a, b, c, d, e, f, g*, рис. 2.

Соответственно, *a* – младший разряд кода, *g* – старший разряд кода.

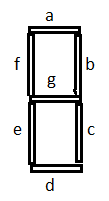


Рис. 2. Разметка сегментов индикатора.

Тогда, как видно на рис.3, «0» нужно записать кодом «1000000»; «1» - кодом «1111001»; «2» - кодом «0100100»;

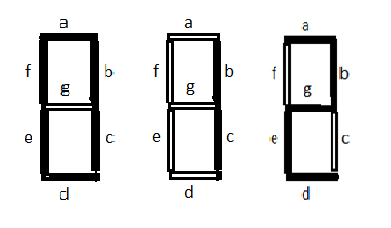


Рис. 3. Примеры отображения символов на семисегментных индикаторах.

Остальные символы, соответственно:

«3» - «0110000»; «4» - «0011001»;

«5» - «0010010»; «6» - «0000010»;

«7» - «1111000»; «8» - «0000000»;

«9» - «0010000»; «*A*» - «0001000»;

«*b*» - «0000011»; «*C*» - «1000110»;

«*d*» - «0100001»; «*E*» - «0000110»;

«*F*» - «0001110».

Программа функционирования устройства записана на языке *Verilog HDL*. Модуль назовем *coder*, его входы обозначим как *data*, а выходы - *seg*. Пользуясь руководством по работе с пакетом *ModelSim*, приведенным ниже, создайте проект, откройте в нем файл *coder*, наберите текст программы 1, сохраните файл и откомпилируйте его.

Программа 1.

*module coder*

(*input wire [*3:0*]* *data*,

*output wire [*6:0*]* *seg*);

*reg* *[*6:0*]code*;

*assign seg=code;*

*always @\**

*case(data)*

4'*b*0000: *code* = 7'*b*1000000;

4*'b*0001: *code* = 7'*b*1111001;

4'*b*0010: *code* = 7'*b*0100100;

4'*b*0011: *code* = 7'*b*0110000;

4'*b*0100: *code* = 7'*b*0011001;

4'*b*0101: *code* = 7'*b*0010010;

4*'b*0110: *code* = 7'*b*0000010;

4*'b*0111: *code* = 7'*b*1111000;

4'*b*1000: *code* = 7'*b*0000000;

4'*b*1001: *code* = 7'*b*0010000;

4'*b*1010: *code* = 7'*b*0001000;

4'*b*1011: *code* = 7'*b*0000011;

4'*b*1100: *code* = 7'*b*1000110;

4'*b*1101: *code* = 7'*b*0100001;

4'*b*1110: *code* = 7*'b*0000110;

4'*b*1111: *code* = 7'*b*0001110;

*endcase*

*endmodule*

Далее приступаем к написанию файла *testbench*. Если в основном модуле входные переменные всегда имеют тип *wire*, то в файле *testbench* на вход модуля поступают сигналы от генератора. Это переменные типа *reg*. Выходные переменные всегда представлены цепями непрерывного назначения, так как это график функции. Тип *wire*.

Модуль *coder* присутствует в общем модуле (рис. 1) *coder\_tb* под меткой *test*; его входы *data* подключены ко входам модуля *coder\_tb - test\_in*, его выходы подключены к выходам модуля *coder\_tb – test\_out*.

Открываем блок *initial* и меняем состояния входов через каждые 200*ns*.

Директива *`timescale* 1*ns*/100*ps* определяет единицу времени для шага симулятора в программе – 1*ns*, а единичный шаг симулятора на графике – 100*ps*.

Запишите текст программы 2 в проект, сохраните файл и откомпилируйте. Далее, следуя указаниям руководства по работе с пакетом *ModelSim*, приступайте к этапу симуляции.

Программа 2.

*`timescale* 1*ns*/100*ps*

*module coder\_tb*;

*reg* *[*3:0*]* *test\_in*;

*wire* *[*6:0*]* *test\_out*;

*coder test (.data(test\_in), .seg(test\_out));*

*initial*

*begin*

*test\_in*=4'*b*0000;

#200;

*test\_in*=4'*b*0001;

#200;

*test\_in*=4'*b*0010;

#200;

*test\_in*=4'*b*0011;

#200;

*test\_in*=4'*b*0100;

#200;

*test\_in*=4'*b*0101;

#200;

*test\_in*=4'*b*0110;

#200;

*test\_in*=4'*b*0111;

#200;

*$stop;*

*end*

*endmodule*

После успешного результата моделирования кодопреобразователя, создайте в том же проекте модуль демультиплексора, программа 3. Самостоятельно напишите файл *testbench* для этого устройства и получите график. Обратите внимание на то, сколько входов и выходов имеет модуль, описанный программой 3.

Программа 3.

*module dms*

*(input wire line,*

*input wire* *[*1:0*]* *adr*,

*output wire data*0,

*output wire data*1,

*output wire data*2,

*output wire data*3 );

*assign data*0 = (*adr*== 2'*h*0)? *line* : 1'*b*0;

*assign data*1 = (*adr*== 2'*h*1)? *line* : 1'*b*0;

*assign data*2 = (*adr*== 2'*h*2)? *line* : 1'*b*0;

*assign data*3 = (*adr*== 2'*h*3)? *line* : 1'*b*0;

*endmodule*

Руководство по работе с пакетом *ModelSim*.

1. Запустить *ModelSim*, используя путь: Приложения – Программирование - *Modelsim* *Altera* *Quartus* 15. Окно информации *ALTERA* закрыть.
2. Создать проект. Для этого в верхнем меню активировать *File – New –**Project.* В окне *Create Project* записать имя проекта - *Project Name* и прописать путь к проекту в своей директории - *Project Location* через поиск. Обязательно проверить наличие по умолчанию библиотеки *work*. Проверить наличие записи исходного пути (*Copy Setting From);/opt/altera*15.0*/modelsim\_ase/modelsim.ini*.
3. Создать новый файл. В открывшемся окне *Add items to the Project*(рис. 4) активировать опцию *Create New File*(рис. 5).

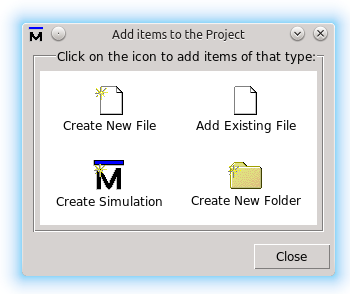


Рис. 4. Окно выбора добавлений в проект.

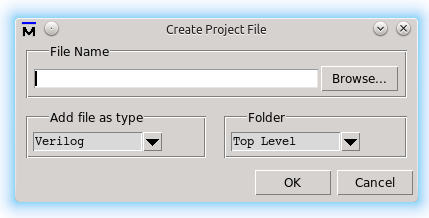


Рис. 5. Окно добавления вновь создаваемого файла.

Записать имя файла в верхней строке, выбрать тип *Verilog*, проверить, что файл верхнего уровня. Если необходимо добавить в проект ранее созданный файл, воспользуйтесь опцией окна *Add items to the Project - Add Existing File*(рис. 6)

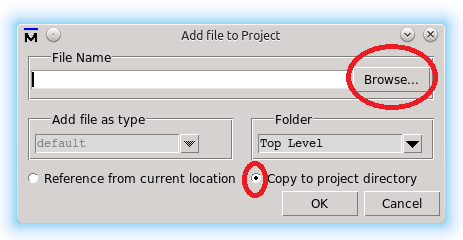


Рис. 6. Окно добавления ранее созданного файла.

Файл находите через поиск и устанавливаете режим копирования.

Те же действия возможно производить, не открывая окно *Add items to the Project*,а два раза кликнув правой клавишей мыши в поле проекта. Откроется маленькое окно, в котором следует активировать строку *Add to Project*.

1. Открыть поле файла через двойной клик по названию. Записать текст, сохранить. Статус сохраненных, но не откомпилированных файлов отмечается синим знаком вопроса, рис. 7.

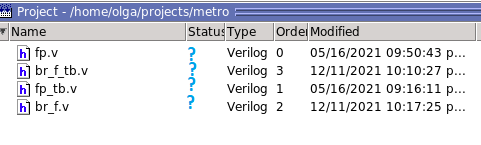


Рис. 7. Обозначение статуса сохраненных в проекте файлов.

1. Откомпилировать. *Compile – Compile All*. Компиляция без ошибок отмечается галочкой зеленого цвета возле имени файла, рис. 8.

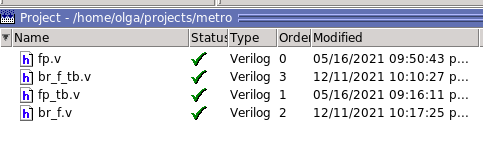


Рис. 8. Обозначение статуса файлов, откомпилированных без ошибок.

Если какой-либо из файлов записан с ошибкой, его статус отмечается красным перекрестом (рис. 9), а под полем проекта следует указание об ошибках. Его можно открыть кликом мыши.

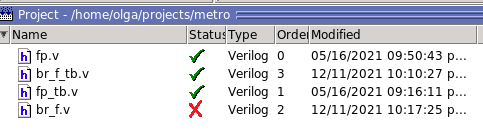


Рис. 9. Статус одного из файлов проекта – ошибка компиляции.

1. После успешной компиляции основного модуля создать новый файл для *testbench*. Для этого кликом правой клавиши мыши в поле под названием файлов вызвать окно и активировать ***Add to Project - New File****.* Записать имя, выбрать тип ***Verilog***. Открыть поле файла, записать текст, сохранить.
2. Откомпилировать все файлы проекта. ***Compile – Compile All***. Компиляция без ошибок отмечается галочкой возле имени файла.
3. При успешной компиляции запустить симуляцию. ***Simulate – Start Simulate*,** рис. 10**.**

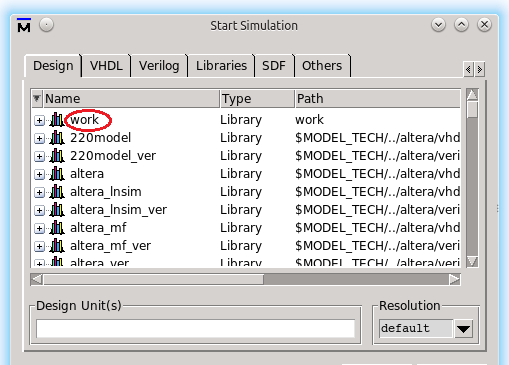
****

Рис. 10. Окно библиотек при старте симуляции.

1. В открывшемся окне библиотек раскрыть библиотеку *work* и отметить файл для симуляции, файл *testbench*. При этом должны открыться окна *sim, Objects* и *Wave*. Из окна *Objects*,рис. 11, перенести название выводов в *Wave*. В примере дан список выводов для конечного автомата.

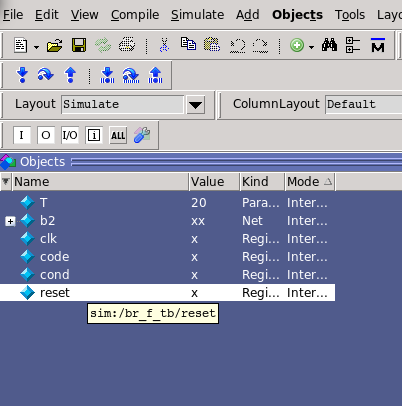


Рис. 11. Вид окна *Objects*.

Для этого выделить их и правой кнопкой мыши нажать *Add Wave*, рис. 12.

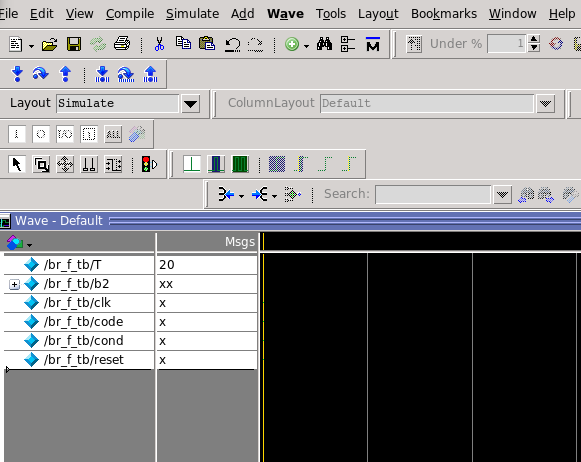
****

Рис. 12. Вид окна *Wave* с перенесенным списком линий.

1. После добавления всех необходимых линий необходимо выполнить построение диаграммы. Для этого используется специальная панель инструментов, находящаяся в верхней части основного окна программы, рис. 13.



Рис. 13. Панель инструментов для запуска симуляции.

Перечислим по порядку элементы панели и их назначение:

* *Restart* – используется для перезапуска симуляции при изменении списка исследуемых линий;
* Текстовое поле – используется для указания интервала времени, для которого необходимо выполнить симуляцию;
* *Run* – используется для выполнения симуляции на заданный интервал времени;
* *ContinueRun* – используется для многократного выполнения симуляции на заданный интервал времени пока не будет достигнута точка останова;
* *Run –All* – используется для выполнения беспрерывной симуляции пока не будет достигнута команда *stop* или *break*;
* *Break* – используется для остановки запущенной симуляции.

Используем третью справа иконку - *Run –All*. Движок под диаграммой установить в крайнее левое положение. Для изменения масштаба можно использовать лупу (+ или -), или горячие клавиши (*O* или *I*). Пример диаграммы для конечного автомата, рис. 14.

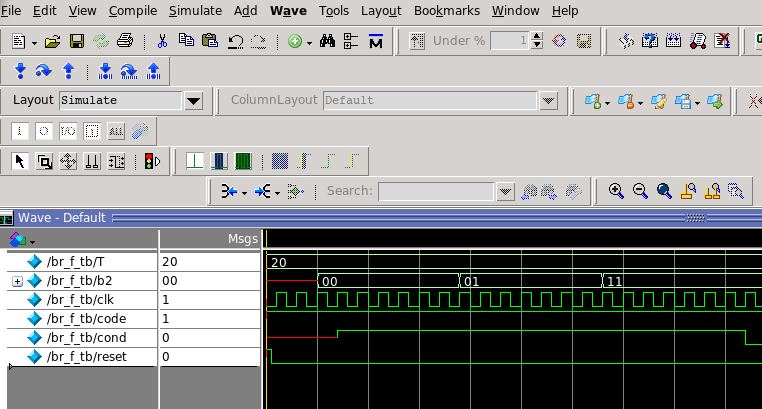


Рис. 14. Полученный график устройства.

После завершения анализа необходимо завершить работу симулятора, для этого выбрать пункт меню *«Simulate» -> «End Simulation».*

Лабораторная работа №1.

**Проектирование двухмодульного КЦУ в *CycloneV.***

**Цель работы.** Изучение методов проектирования линейных схем в ПЛИС с использованием пакетов *Modelsim* и *Quartus*15.

Задание на работу в лаборатории. Спроектировать устройство, преобразующее 4-разрядные данные, подаваемые с тумблеров макета *DE*1-*SoC* в код для семисегментного индикатора, с возможностью вывода числа на один из четырех индикаторов макета, выбираемого по вводимому с тумблеров адресу, рис 1.1.

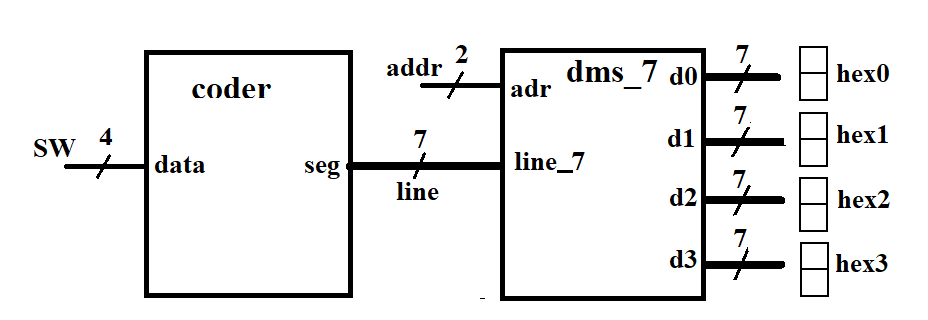


Рис. 1.1. Схема проектируемого устройства.

Порядок выполнения работы.

1. Открыть новый проект в *Modelsim*. Скопировать в него файлы кодера и демультиплексора, созданные на предыдущих занятиях.
2. Изменить файл демультиплексора, превратив устройство в 7 объединенных демультиплексоров, подключенных параллельным образом по адресной шине. Сохранить файл в проекте с новым именем, откомпилировать файлы.
3. Написать *testbench* для файла построения 7-ми демультиплексоров, откомпилировать, провести симуляцию, проверить ее результат.
4. Записать результирующий файл устройства, состоящего из модулей кодера и подключенному к его выходу модулю 7 демультиплексоров. Сборка производится по тому же принципу, как и включение модуля в файле *testbench*. Двухмодульное устройство будет иметь две строки сборки: для монтирования модуля кодера и для монтирования модуля демультиплексора. Откомпилировать файл.
5. Записать *testbench* для результирующего файла, откомпилировать, провести симуляцию, проверить ее результат.
6. При положительном результате функциональной симуляции перенести файлы основных модулей проекта в *Quartus*15. Для этого, пользуясь Приложением 1.1, открыть *Quartus*15, создать в нем новый проект, добавив в него при создании 3 файла: файлы кодера, 7-ми демультиплексоров и результирующий.
7. Откомпилировать все файлы в *Quartus*15. Можно компилировать результирующий, поставив его старшим в иерархии, так как он опирается на два остальных модуля. Компиляцию проводить по полной программе!
8. Пользуясь Приложением 1.2 произвести разводку выводов схемы для работы в макете таким образом, чтобы ввод числа осуществлялся с тумблеров *SW*9, *SW*8, *SW*7, *SW*6(*SW*9 – старший разряд), ввод адреса – с тумблеров *SW*1, *SW*0(*SW*1 – старший разряд). Вывод производить на сегментные индикаторы с 0-го по 3-ий. После компиляции файла планировщика еще раз откомпилируйте файл верхнего уровня!
9. Пользуясь Приложением 1.3 произвести программирование кристалла *FPGA* макета.

Проверить работу устройства**.** Порядок проверки следующий: на выделенных под адреса и данные тумблерах устанавливаем адрес, устанавливаем любое число. Убедившись, что вывод производится, устанавливаем новый адрес и повторяем вывод любого числа.

Продемонстрировать работу преподавателю.

**Отчет должен содержать программы функционирования устройств и диаграммы их работы.**

Лабораторная работа №2.

**Проектирование элемента АЛУ в *CycloneV*.**

**Цель работы.** Изучение методов проектирования тактируемых схем в ПЛИС с использованием пакетов *Modelsim* и *Quartus*15.

**Задание на работу в лаборатории**. Спроектировать устройство, позволяющее производить арифметическое сложение 4-разрядных данных, подаваемых с тумблеров макета *DE*1-*SoC*, сохраняя результат в регистре, а бит переноса в триггере, представленном, как старший разряд того же регистра. Полученная сумма выводится на семисегментный индикатор, а состояние переноса – на светодиод макета, рис. 2.1.

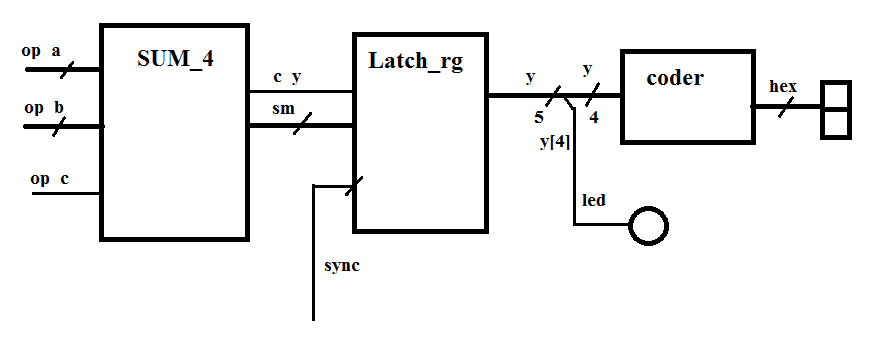


Рис. 2.1. Блок-схема реализуемого устройства.

Порядок выполнения работы.

1. Открыть новый проект в *Modelsim*. Скопировать в него файл кодера, созданный на предыдущих занятиях.
2. Открыть новый файл и записать в него текст кода одноразрядного сумматора, программа 2.1. Сохранить и скомпилировать файл.

Программа 2.1.

*module sum*

*(input wire a,b,cr,*

*output wire s,crp);*

*assign s=(a^b)^cr;*

*assign crp=(a&b) | ((a^b)&cr);*

*endmodule*

1. Написать *testbench* для файла одноразрядного сумматора, откомпилировать, провести симуляцию, проверить ее результат.
2. Добавить в проект новый файл для 4-разрядного сумматора, программа 2.2. Записать текст, сохранить, откомпилировать файл.

Программа 2.2.

*module* *sum\_*4

(*input* *wire* *[*3:0*]a\_in*,*b\_in*,

*input* *wire* *cr\_in*,

*output* *wire* *[*3:0*]s\_out*,

*output* *wire* *crp\_out*);

*wire* *[*2:0*]crp\_n*;

*sum* *sum*0(.*a* (*a\_in[*0*]*),.*b* (*b\_in[*0*]*),.*cr*(*cr\_in*),.*s*(*s\_out[*0*]*),.*crp*(*crp\_n[*0*]*));

*sum* *sum*1(.*a*(*a\_in[*1*]*),.*b*(*b\_in[*1*]*),.*cr*(*crp\_n[*0*]*),.*s*(*s\_out[*1*]*),.*crp*(*crp\_n[*1*]*));

*sum* *sum*2(.*a* (*a\_in[*2*]*),.*b* (*b\_in[*2*]*),.*cr*(*crp\_n[*1*]*),.*s*(*s\_out[*2*]*),.*crp*(*crp\_n[*2*]*));

*sum* *sum*3(.*a* (*a\_in[*3*]*),.*b* (*b\_in[*3*]*),.*cr*(*crp\_n[*2*]*),.*s*(*s\_out[*3*]*),.*crp*(*crp\_out*));

*endmodule*

1. Написать *testbench* для файла 4-разрядного сумматора, откомпилировать, провести симуляцию, проверить ее результат.
2. Добавить в проект новый файл для 5-разрядного регистра-защелки, программа 2. 3. Записать текст, сохранить, откомпилировать файл.

Программа 2.3.

*module* *latch\_rgstr*

#(*parameter* *N*=5)

(*input* *[N*-1:0*]* *d\_in*,

*input* *clk*,

*output* *[N*-1:0*]d\_out*);

*reg* *[N*-1:0*]q*;

*assign* *d\_out*=*q*;

*always*@ (*posedge* *clk*)

*begin*

*q*<=*d\_in*;

*end*

*endmodule*

1. Написать *testbench* для файла 5-разрядного регистра-защелки, программа 2. 4, откомпилировать, провести симуляцию, проверить ее результат. При составлении файла *testbench* необходимо учитывать, что регистр записывает информацию на фронте импульса тактовой частоты. Для этого необходимо установить параметризованную величину периода и задать скважность частоты в блоке *always*. Но подавать данные на входы регистра необходимо до возникновения фронта синхронизации. Поэтому каждое новое значение данных следует подавать под спад (отрицательный край) частотного импульса.

Программа 2. 4.

*`timescale* 1*ns*/10*ps*

*module* *test\_lr\_*5;

*localparam* *T*=20;

*reg* *clk*;

*reg* *[*4:0*]in\_d*;

*wire* *[*4:0*]out\_d*;

*latch\_rgstr* *test*(.*clk*(*clk*),.*d\_in*(*in\_d*),.*d\_out*(*out\_d*));

*always*

*begin*

*clk*=1'*b*0;

#(*T*/2);

*clk*=1'*b*1;

#(*T*/2);

*end*

*initial*

*begin*

*in\_d*=5'*b*10101;

@(*negedge* *clk*);

*in\_d*=5'*b*01100;

@(*negedge* *clk*);

*in\_d*=5'*b*11001;

@(*negedge* *clk*);

*in\_d*=5'*b*01011;

@(*negedge* *clk*);

$*stop*;

*End*

*endmodule*

1. Записать результирующий файл устройства, состоящего из модулей 4-разрядного сумматора, регистра-защелки и кодера. Откомпилировать файлы.

*module* *summ\_*4

(*input* *wire* *[*3:0*]op\_a*,*op\_b*,

*input* *wire* *op\_c*, *sync*,

*output* *wire* *led*,

*output* *wire* *[*6:0*]hex*);

*wire* *[*3:0*]sm*;

*wire* *c\_y*;

*wire* *[*4:0*]y*;

*assign* *led*=*y[*4*]*;

*sum\_*4 *block*1(.*a\_in*(*op\_a*), .*b\_in*(*op\_b*), .*cr\_in*(*op\_c*), .*s\_out*(*sm*), .*crp\_out*(*c\_y*));

*latch\_rgstr* *block*2(.*d\_in*({*c\_y*,*sm[*3:0*]*}), .*clk*(*sync*), .*d\_out*(*y*));

*coder* *block*3(.*data*(*y[*3:0*]*), .*seg*(*hex*));

*endmodule*

1. Записать *testbench* для результирующего файла, откомпилировать, провести симуляцию, проверить ее результат.
2. При положительном результате функциональной симуляции перенести файлы основных модулей проекта в *Quartus*15. Для этого открыть *Quartus*15, создать в нем новый проект, добавив в него при создании 5 файлов: файлы сумматоров, регистра-защелки, кодера и результирующий.
3. Откомпилировать все файлы в *Quartus*15. Можно компилировать результирующий, поставив его старшим в иерархии, так как он опирается на все входящие в него модули. Компиляцию проводить по полной программе!
4. Пользуясь Приложением 1.2 произвести разводку выводов схемы для работы в макете таким образом, чтобы ввод чисел осуществлялся с тумблеров *SW*9 - *SW*6(*SW*9 – старщий разряд) – первое слагаемое, *SW*5 –*SW*2 - второе слагаемое; вход переноса – с тумблера *SW*0. Частотаподается с кнопки *KEY*(0). Вывод суммы производить на 1-ый сегментный индикатор. Вывод переноса на светодиод *LEDR*1. После компиляции файла планировщика еще раз откомпилируйте файл верхнего уровня!
5. Пользуясь Приложением 1.3 произвести программирование кристалла *FPGA* макета.

Проверить работу устройства**.** Порядок проверки следующий: устанавливаем любые числа на тумблерах. Нажимаем кнопку крайнюю справа. Проверяем результат.

Продемонстрировать работу преподавателю.

**Отчет должен содержать:**

1. **цель работы;**
2. **блок-схему устройства, описанного в результирующем файле;**
3. **программы функционирования каждого блока и устройства в целом;**
4. **Резюме: «функционирование устройства поблочно проверено с помощью пакета *Modelsim*, устройство реализовано в макете *DE*1-*SoC* с использованием пакета *Quartus* 15».**

Лабораторная работа *N*3.

**Проектирование двоичного счетчика в *CycloneV*.**

**Цель работы.** Изучение методов проектирования простых конечных автоматов в ПЛИС с использованием пакетов *Modelsim* и *Quartus*15.

**Задание на работу в лаборатории.** Спроектировать 4-разрядный реверсивный счетчик с возможностями асинхронного и синхронного сбросов, синхронной загрузки состояния и с разрешением счета, рис. 3.1. Частота подается на реверсивный счетчик от генератора через понижающий счетчик. Вывод состояний счетчика производится на 7-сегментный индикатор. Управляющие сигналы подаются с тумблеров макета. Загружаемое состояние также вводится с тумблеров.

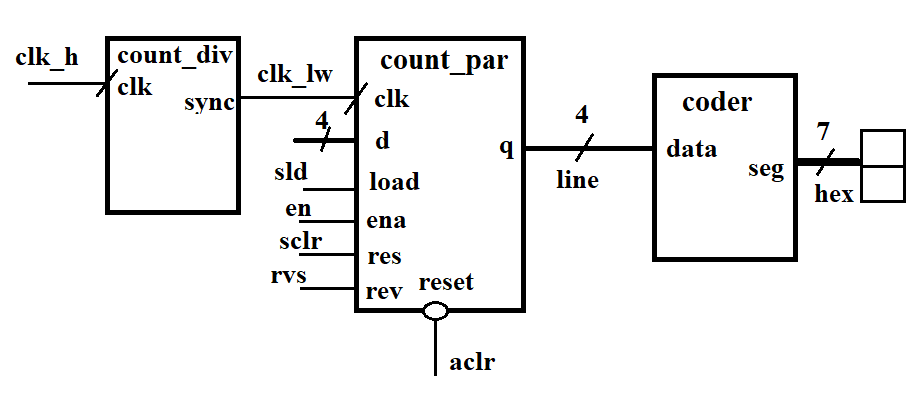


Рис. 1.3. Блок-схема реализуемого устройства.

**Порядок выполнения работы.**

1. Открыть новый проект в *Modelsim*. Скопировать в него файл кодера, созданный на предыдущих занятиях.

2. Добавить новый файл реверсивного счетчика, записать текст модуля, программа 3.1, сохранить и откомпилировать файл.

Программа 3. 1.

*module* *count\_par*

# (*parameter* *N*=4)

(*input* *wire* *clk*,*reset*,*res*,*ena*,*rev*,*load*,

*input* *wire* *[N*-1:0*]d*,

*output* *wire* *[N*-1:0*]q*);

*reg* *[N*-1:0*]cnt*;

*reg* *[N*-1:0*]cnt\_next*;

*assign* *q*=*cnt*;

*always*@(*posedge* *clk*, *negedge* *reset*)

*if*(!*reset*)

*cnt*<=0;

*else*

*cnt*<=*cnt\_next*;

*always*@\*

*if*(*res*)

*cnt\_next*=0;

*else* *if*(*load*)

*cnt\_next*<=*d*;

*else* *if*(*ena*&!*rev*)

*cnt\_next*=*cnt* + 1;

*else* *if*(*ena*&*rev*)

*cnt\_next*=*cnt*-1;

*else*

*cnt\_next*=*cnt*;

*endmodule*

3. Написать *testbench* для реверсивного счетчика, откомпилировать, провести симуляцию, проверить ее результат. При написании файла *testbench* необходимо учесть, что сигнал асинхронного сброса удерживается активным только в начале работы устройства. Поэтому описание действия *reset* в *testbench* представляется рациональным вынести в отдельный блок *initial*, сразу после распределения скважности частотного сигнала, например

*initial*

*begin*

*reset\_tb*=1’*b*0;

#(*T*/4);

*reset\_tb*=1’*b*1;

*end*

4. Добавить новый файл счетчика для понижения частоты, записать текст модуля, программа 3. 2, сохранить и откомпилировать файл.

Программа 3. 2.

*module* *count\_div*2

# (*parameter* *N*=4,*M*=4'*d*10)

(*input* *wire* *clk*,

*output* *reg* *sync*=0,

*output* *wire* *[N*-1:0*]q*);

*reg* *[N*-1:0*]cnt*=0;

*wire* *[N*-1:0*]cnt\_next*;

*assign* *q*=*cnt*;

*assign* *cnt\_next*=*cnt*+1'*b*1;

*always*@(*posedge* *clk*)

*begin*

*cnt*<=*cnt\_next*;

*sync*<=*sync*;

*if* (*cnt*==*M*-1)

*begin*

*cnt*<=0;

*sync*<=~*sync*;

*end*

*end*

*endmodule*

5. Написать *testbench* для понижающего счетчика, откомпилировать, провести симуляцию, проверить ее результат.

6. Записать результирующий файл устройства, состоящего из модулей понижающего счетчика, реверсивного счетчика и кодера. Откомпилировать файл.

7. Записать *testbench* для результирующего файла, откомпилировать, провести симуляцию, проверить ее результат.

8. При положительном результате функциональной симуляции перенести файлы основных модулей проекта в *Quartus*15.

9. Откомпилировать все файлы в *Quartus*15. Можно компилировать результирующий, поставив его старшим в иерархии. Компиляцию проводить по полной программе!

10. Пользуясь Приложением 1.2 произвести разводку выводов схемы для работы в макете таким образом, чтобы ввод загружаемого состояния осуществлялся с тумблеров

*SW*9, *SW*8, *SW*7, *SW*6, ввод управления: – с тумблера *SW*0 – асинхронный сброс; *SW*1 – синхронный сброс; *SW*2 – разрешение счета; *SW*3 – реверс; *SW*4 - загрузка. Вывод

производить на сегментный индикатор по выбору. После компиляции файла планировщика необходимо снова откомпилировать файл верхнего уровня!

Пользуясь Приложением 1.3 произвести программирование кристалла *FPGA* макета.

Проверить работу устройства. Порядок установки управлений должен соответствовать коду файла *testbench*.

Продемонстрировать работу преподавателю.

**Отчет должен содержать:**

1. **цель работы;**
2. **блок-схему устройства, описанного в результирующем файле;**
3. **программы функционирования каждого блока и устройства в целом;**
4. **Резюме: «функционирование устройства поблочно проверено с помощью пакета *Modelsim*, устройство реализовано в макете *DE*1-*SoC* с использованием пакета *Quartus* 15».**

Лабораторная работа №4.

**Проектирование сдвигового регистра.**

**Цель работы**: изучение методов проектирования устройств, основанных на регистрах сдвига.

**Задание на работу в лаборатории**.

Необходимо получить последовательный вывод чисел, набираемых с тумблеров макета, на шесть 7-сегментных индикаторов. Вывод должен производиться с частотой 1 Гц. (Тактовая частота кристалла в макете 50МГц). Блок-схема устройства с указанием типа используемых переменных представлена на рис. 4. 1.

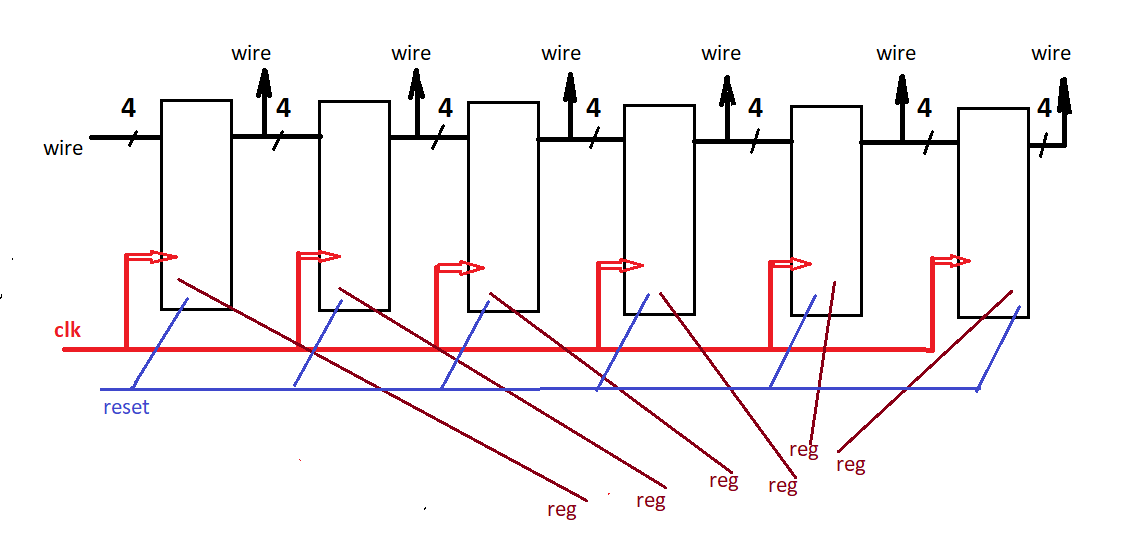
****

Рис. 4. 1. Блок-схема реализуемого устройства.

Ниже приведен пример программирования регистра сдвига на одноразрядных ячейках, рис. 4. 2.

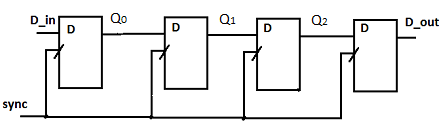


Рис. 4. 2. Регистр сдвига на 4-х одноразрядных ячейках.

*module* *shift\_reg*

#(*parameter* *N*=4)

(*input* *clk*, *d\_in*,

*output* *d\_out*);

*reg* *[N*-1:0*]* *d*;

*assign* *d\_out* = *d[N*-1*]*;

*always* @ (*posedge* *clk*)

*begin*

*d[N*-1:1*]* <= *d[N*-2:0*]*;

*d[*0*]* <= *d\_in*;

*end*

*endmodule*

**Порядок выполнения работы.**

* + - 1. По заданному фрагменту-образцу записать модуль сдвигающего регистра, необходимый для построения устройства. Графическое соответствие образца и задания, рис. 4. 3.

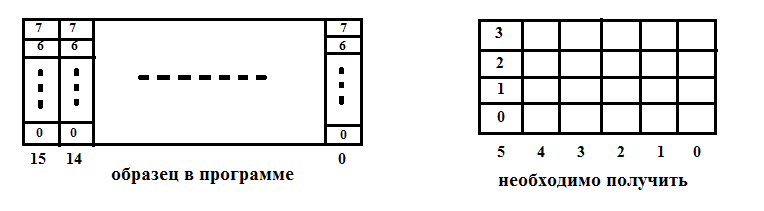


Рис. 4. 3. Графическое соответствие образца и задания.

Описание фрагментов образца.

(*input* *[*7:0*]* *data\_in*,

*input* *clk*,

…….);

*integer* *i*;

*reg* *[*7:0*]* *shift\_reg* *[*15:0*]*;

*assign* ……….

*always*@(*posedge* *clk*, *posedge* *reset*)

*begin*

*if* (*reset*)

*begin*

*for* (*i*=0; *i*<16; *i*=*i*+1)

…………..

*end*

*else*

*begin*

*for* (*i*=1; *i*<16; *i*=*i*+1)

*shift\_reg[i]* <= *shift\_reg[i*-1*]*

*shift\_reg[*0*]* <= *data\_in*;

*end*

……..

* + - 1. Записать *testbench* для созданного модуля.
      2. Расписать, какие еще модули понадобятся для построения устройства, и в каких количествах; обозначить названия входов и выходов модулей.
      3. Записать код результирующего модуля.
      4. Написать *testbench* для результирующего модуля.
      5. Произвести макетирование устройства в *CycloneV*. Для вывода набирать:

а) Ряд целых чисел: 1, 2, 3, 4, 5, 6, 7…

б) Фразу *HELLO* *FRIEnd* (заменить кодопреобразователь).

**Результаты продемонстрировать преподавателю.**

**Отчет должен содержать:**

1. **цель работы;**
2. **блок-схему устройства, описанного в результирующем файле;**
3. **программы функционирования каждого блока и устройства в целом;**
4. **Резюме: «функционирование устройства поблочно проверено с помощью пакета *Modelsim*, устройство реализовано в макете *DE*1-*SoC* с использованием пакета *Quartus* 15».**

Лабораторная работа *N* 5.

**Программное проектирование конечного автомата.**

**Цель работы**. Изучение методов программного проектирования конечных автоматов по рассмотренному примеру задания на проектирование. Определение количества состояний автомата, условий перехода и действий внутри состояния.

**Задание на работу в лаборатории.**

Записать программу функционирования конечного автомата, который, в зависимости от состояния управляющего бита, считывает числа с одного из портов (*a* или *b*) и выводит их на семи-сегментные индикаторы. В каждом цикле автомат ожидает подготовку данных в течение 3-х тактов. После вывода полученных данных автомат обнуляет счетчик тактов и уходит в режим ожидания. Самостоятельно изобразить граф переключений автомата.

Порядок выполнения работы.

1. В пакете *ModelSim* создать проект, записать код Программы 1 для конечного автомата, откомпилировать.
2. Составить *testbench* для созданного файла, откомпилировать и проверить на функциональной диаграмме. 3. Записать код результирующего файла. Изобразить устройство блочно; в понижающем счетчике установить понижающий коэффициент 4.
3. Составить *testbench* для созданного результирующего файла, откомпилировать и проверить на функциональной диаграмме.
4. Перенести файлы проекта в *Quartus*, откомпилировать проект, произвести моделирование устройства в кристалле макета по методике предыдущих работ.

Программа 1.

*module* *moore\_lab*

(*input* *clk*,*flag*,

*input* *reset*,

*input* *[*2:0*]* *a*,

*input* *[*2:0*]* *b*,

*output* *reg* *[*2:0*]* *out*);

*reg* *[*2:0*]* *state*;

*reg* *[*1:0*]* *cnt*;

*parameter* *Res* = 0, *Weit* = 1, *Flag\_read*= 2, *A\_out* = 3, *B\_out* = 4, *Fin* = 5;

*always* @ (*posedge* *clk* *or* *posedge* *reset*)

*begin*

*if* (*reset*)

*state* <= *Res*;

*else*

*begin*

*case* (*state*)

*Res*:

*state* <= *Weit*;

*Weit*:

*if*(*cnt*==2'*d*3)

*state* <= *Flag\_read*;

*Flag\_read*:

*if*(!*flag*)

*state* <= *A\_out*;

*else*

*state* <= *B\_out*;

*A\_out*:

*if*(*cnt*==2'*d*2)

*state* <= *Fin*;

*B\_out*:

*if*(*cnt*==2'*d*2)

*state* <= *Fin*;

*Fin*:

*state* <= *Weit*;

*default*:

*state* <= *Res*;

*endcase*

*end*

*end*

*always* @ (*posedge* *clk*)

*begin*

*case* (*state*)

*Res*:

*begin*

*cnt* <= 2'*d*0;

*out* <= 3'*b*000;

*end*

*Weit*:

*begin*

*if* (*cnt*==2'*d*3)

*cnt* <= 2'*d*0;

*else*

*cnt* <= *cnt* + 2'*d*1;

*out* <= 3'*b*000;

*end*

*A\_out*:

*begin*

*if* (*cnt*==2'*d*2)

*cnt* <= 2'*d*0;

*else*

*cnt* <= *cnt* + 2'*d*1;

*out* <= *a*;

*end*

*B\_out*:

*begin*

*if* (*cnt*==2'*d*2)

*cnt* <= 2'*d*0;

*else*

*cnt* <= *cnt* + 2'*d*1;

*out* <= *b*;

*end*

*endcase*

*end*

*endmodule*

Продемонстрировать работу устройства преподавателю.

**Отчет должен содержать:**

1. **цель работы;**
2. **граф переключений конечного автомата;**
3. **блок-схему устройства, описанного в результирующем файле;**
4. **программы функционирования каждого блока и устройства в целом;**
5. **Резюме: «функционирование устройства поблочно проверено с помощью пакета *Modelsim*, устройство реализовано в макете *DE*1-*SoC* с использованием пакета *Quartus* 15».**

Лабораторная работа *No* 6.

**Программное проектирование конечного автомата.**

**Цель работы**. Изучение методов программного проектирования конечных автоматов по выданному заданию на проектирование. Определение количества состояний автомата, условий перехода и действий внутри состояния.

**Задание на работу в лаборатории.**

Задание выдается преподавателем. Варианты заданий приведены в приложении 2. Каждое задание может быть выдано как для индивидуальной работы, так и для работы в бригаде из 2-х студентов.

Приложение 1.

**Работа с пакетом *Quartus*15.**

Пакет *Quartus* предназначен для макетирования устройств в кристаллах ПЛИС. В лаборатории кафедры ПИВТ изучаются возможности макетирования в *FPGA* *CycloneV* в составе *DE*1-*SoC*. Для этих целей используется версия пакета *Quartus* 15. В приложении даются основные описания привязки выводов схемы к периферии и необходимые сведения о работе в различных разделах пакета.

П. 1. 1. Создание проекта.

1. Активируйте значок запуска или произведите поиск по пути: **приложения - программирование – *quartus*15**. При первом входе вы должны выбрать лицензию, которая была выдана вместе с пакетом. В открывшемся окне ***Start* *Designing***  выберите ***New* *Project* *Wizard***. (Если Вы собираетесь открывать уже имеющийся проект, то ***Open* *Project***).
2. Откроется окно ***Introduction***, нажмите ***Next***.
3. Откроется окно ***Directory*, *Name*, *Top*-*Level* *Entity***. В первой строке найдите Вашу папку пользователя, где будут находиться проекты. Во второй строке запишите имя проекта. В третьей строке появится имя файла верхнего уровня, совпадающее с именем проекта. Нажмите ***Next***. (При повторных входах Вам объявят, что в данной папке уже есть проекты, и предложат поменять папку. Этого делать не следует. Ответьте ***No***).
4. Следующее окно ***Project* *Type***.Оставьте ***Empty* *project***. Нажмите ***Next***.
5. Следующее окно ***Add* *Files***. Если это первый вход в пакет, у Вас еще нет файлов, которые можно добавить в проект. Если Вы уже работали, то в строке ***File* *name*** найдите, активируя ***[*…*]***, нужный файл и добавьте его в проект, нажав ***Add***. Нажмите ***Next***.
6. Последнее окно позволяет Вам выбрать тип микросхемы**. *Family* & *Device* *Setting*. *Family* *CycloneV*(…). *Device* *CycloneVSE* *Mainstream*. *Name* 5*CSEMA*5*F*31*C*6**. Нажмите  ***Finish***.
7. Открылось окно проекта. Во второй строке верхнего меню должно присутствовать название проекта, справа от названия должна быть активна кнопка навигатора проекта, рис. П1. 1.



Рис. П1. 1. Вид кнопки навигатора проекта.

1. Пользуясь первой строкой верхнего меню, выберите ***File* – *New* – *VerilogHDL* *File***. В открывшееся поле запишите текст программы. Слева от поля текста программы должно быть окно навигатора. Активируйте кнопку «Файлы», рис. П 1. 2.

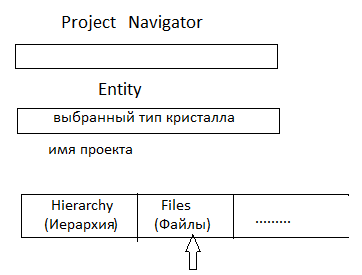


Рис. П1. 2. Выбор кнопки «Файлы» в окне навигатора.

1. После набора текста нажмите ***File* – *Save* *as*** и сохраните файл под тем именем, которое дано ему в программе в блоке ***module***. Вы увидите это название в окне навигатора. Выделите его правой клавишей мыши и, в открывшемся окне выберите ***Set* *as* *top*-*level* *entity***.
2. Запустите файл на компиляцию: ***Processing* – *Start* *Compilation*** или значок – сиреневый треугольник на второй строке верхнего меню. Внимательно следите за этапами компиляции. Они просматриваются слева, ниже окна навигатора. Ошибки выделяются красным цветом. Если сообщения об ошибках появились в нижнем окне сообщений, найдите первое и постарайтесь его прочитать. Откройте файл программы через его название в навигаторе и дважды кликните строку ошибки. В программе будет выделена строка с ошибочной записью. Постарайтесь исправить ошибки самостоятельно, снова сохранить текст файла и поставить его на компиляцию. В случае удачного результата переходите к программе планировщика.

П. 1. 2. Планировщик.

Для программирования устройства в кристалл макета необходимо связать внутренние входы и выходы с контактными площадками выводов микросхемы. Чтобы выполнить это, воспользуйтесь файлом планировщика.

Активизация файла: ***Assignment* – *Pin* *Planer*.**

Проверьте тип кристалла: **5*CSEMA*5*F*31*C*6**.

В открывшейся таблице заполните графу ***Location*.**

Поле таблицы активируется двумя кликами мыши в позиции графы ***Location*** напротив выбранного входа или выхода схемы, после чего в обозначенную позицию вписывается номер контакта.

Номера контактов выбирайте из таблиц данного приложения.

После заполнения таблицы производится компиляция внешней части кристалла по пути: ***Processing* – *Start* *I*/*O* *Assignment* *Analysis*.**

**По завершении анализа и синтеза внешней части проведите полную компиляцию проекта, объединив, таким образом, внутреннюю и внешнюю части в единое целое.**

**Выводы тактовой частоты.**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table* 3-5*Pin* *Assignments* *for* *Clock* *InputsSignal Name*** | ***FPGA Pin No.*** | ***Description*** | ***I/O Standard*** |
| ***CLOCK\_*50** | ***PIN\_AF*14** | **50 *MHz* *clock* *input*** | **3.3*V*** |
| ***CLOCK*2*\_*50** | ***PIN\_AA*16** | **50 *MHz* *clock* *input*** | **3.3*V*** |
| ***CLOCK*3*\_*50** | ***PIN\_Y*26** | **50 *MHz* *clock* *input*** | **3.3*V*** |
| ***CLOCK*4*\_*50** | ***PIN\_K*14** | **50*MHz* *clock* *input*** | **3.3*V*** |
| ***HPS\_CLOCK*1*\_*25** | ***PIN\_D*25** | **25*MHz* *clock* *input*** | **3.3*V*** |
| ***HPS\_CLOCK*2*\_*25** | ***PIN\_F*25** | **25*MHz* *clock* *input*** | **3.3*V*** |

**Тумблеры**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table* 3-6*Pin* *Assignments* *for* *Slide* *SwitchesSignal Name*** | ***FPGA Pin No.*** | ***Description*** | ***I/O Standard*** |
| ***SW[*0*]*** | ***PIN\_AB*12** | ***Slide* *Switch[*0*]*** | **3.3*V*** |
| ***SW[*1*]*** | ***PIN\_AC*12** | ***Slide* *Switch[*1*]*** | **3.3*V*** |
| ***SW[*2*]*** | ***PIN\_AF*9** | ***Slide* *Switch[*2*]*** | **3.3*V*** |
| ***SW[*3*]*** | ***PIN\_AF*10** | ***Slide* *Switch[*3*]*** | **3.3*V*** |
| ***SW[*4*]*** | ***PIN\_AD*11** | ***Slide* *Switch[*4*]*** | **3.3*V*** |
| ***SW[*5*]*** | ***PIN\_AD*12** | ***Slide* *Switch[*5*]*** | **3.3*V*** |
| ***SW[*6*]*** | ***PIN\_AE*11** | ***Slide* *Switch[*6*]*** | **3.3*V*** |
| ***SW[*7*]*** | ***PIN\_AC*9** | ***Slide* *Switch[*7*]*** | **3.3*V*** |
| ***SW[*8*]*** | ***PIN\_AD*10** | ***Slide* *Switch[*8*]*** | **3.3*V*** |
| ***SW[*9*]*** | ***PIN\_AE*12** | ***Slide* *Switch[*9*]*** | **3.3*V*** |

**Кнопки**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table* 3-7*Pin* *Assignments* *for* *Push*-*buttonsSignal Name*** | ***FPGA Pin No.*** | ***Description*** | ***I/O Standard*** |
| ***KEY[*0*]*** | ***PIN\_AA*14** | ***Push*-*button[*0*]*** | **3.3*V*** |
| ***KEY[*1*]*** | ***PIN\_AA*15** | ***Push*-*button[*1*]*** | **3.3*V*** |
| ***KEY[*2*]*** | ***PIN\_W*15** | ***Push*-*button[*2*]*** | **3.3*V*** |
| ***KEY[*3*]*** | ***PIN\_Y*16** | ***Push*-*button[*3*]*** | **3.3*V*** |

**Светодиоды**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table* 3-8*Pin* *Assignments* *for* *LEDsSignal Name*** | ***FPGA Pin No.*** | ***Description*** | ***I/O Standard*** |
| ***LEDR[*0*]*** | ***PIN\_V*16** | ***LED* *[*0*]*** | **3.3*V*** |
| ***LEDR[*1*]*** | ***PIN\_W*16** | ***LED* *[*1*]*** | **3.3*V*** |
| ***LEDR[*2*]*** | ***PIN\_V*17** | ***LED* *[*2*]*** | **3.3*V*** |
| ***LEDR[*3*]*** | ***PIN\_V*18** | ***LED* *[*3*]*** | **3.3*V*** |
| ***LEDR[*4*]*** | ***PIN\_W*17** | ***LED* *[*4*]*** | **3.3*V*** |
| ***LEDR[*5*]*** | ***PIN\_W*19** | ***LED* *[*5*]*** | **3.3*V*** |
| ***LEDR[*6*]*** | ***PIN\_Y*19** | ***LED* *[*6*]*** | **3.3*V*** |
| ***LEDR[*7*]*** | ***PIN\_W*20** | ***LED* *[*7*]*** | **3.3*V*** |
| ***LEDR[*8*]*** | ***PIN\_W*21** | ***LED* *[*8*]*** | **3.3*V*** |
| ***LEDR[*9*]*** | ***PIN\_Y*21** | ***LED* *[*9*]*** | **3.3*V*** |

**Семисегментные индикаторы**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table* 3-9*Pin* *Assignments* *for* 7-*segment* *DisplaysSignal Name*** | ***FPGA Pin No.*** | ***Description*** | ***I/O Standard*** |
| ***HEX*0*[*0*]*** | ***PIN\_AE*26** | ***Seven* *Segment* *Digit* 0*[*0*]*** | **3.3*V*** |
| ***HEX*0*[*1*]*** | ***PIN\_AE*27** | ***Seven* *Segment* *Digit* 0*[*1*]*** | **3.3*V*** |
| ***HEX*0*[*2*]*** | ***PIN\_AE*28** | ***SevenSegment* *Digit* 0*[*2*]*** | **3.3*V*** |
| ***HEX*0*[*3*]*** | ***PIN\_AG*27** | ***Seven* *Segment* *Digit* 0*[*3*]*** | **3.3*V*** |
| ***HEX*0*[*4*]*** | ***PIN\_AF*28** | ***Seven* *Segment* *Digit* 0*[*4*]*** | **3.3*V*** |
| ***HEX*0*[*5*]*** | ***PIN\_AG*28** | ***Seven* *Segment* *Digit* 0*[*5*]*** | **3.3*V*** |
| ***HEX*0*[*6*]*** | ***PIN\_AH*28** | ***Seven* *Segment* *Digit* 0*[*6*]*** | **3.3*V*** |
| ***HEX*1*[*0*]*** | ***PIN\_AJ*29** | ***Seven* *Segment* *Digit* 1*[*0*]*** | **3.3*V*** |
| ***HEX*1*[*1*]*** | ***PIN\_AH*29** | ***Seven* *Segment* *Digit* 1*[*1*]*** | **3.3*V*** |
| ***HEX*1*[*2*]*** | ***PIN\_AH*30** | ***Seven* *Segment* *Digit* 1*[*2*]*** | **3.3*V*** |
| ***HEX*1*[*3*]*** | ***PIN\_AG*30** | ***Seven* *Segment* *Digit* 1*[*3*]*** | **3.3*V*** |
| ***HEX*1*[*4*]*** | ***PIN\_AF*29** | ***Seven* *Segment* *Digit* 1*[*4*]*** | **3.3*V*** |
| ***HEX*1*[*5*]*** | ***PIN\_AF*30** | ***Seven* *Segment* *Digit* 1*[*5*]*** | **3.3*V*** |
| ***HEX*1*[*6*]*** | ***PIN\_AD*27** | ***Seven* *Segment* *Digit* 1*[*6*]*** | **3.3*V*** |
| ***HEX*2*[*0*]*** | ***PIN\_AB*23** | ***Seven* *Segment* *Digit* 2*[*0*]*** | **3.3*V*** |
| ***HEX*2*[*1*]*** | ***PIN\_AE*29** | ***Seven* *Segment* *Digit* 2*[*1*]*** | **3.3*V*** |
| ***HEX*2*[*2*]*** | ***PIN\_AD*29** | ***Seven* *Segment* *Digit* 2*[*2*]*** | **3.3*V*** |
| ***HEX*2*[*3*]*** | ***PIN\_AC*28** | ***Seven* *Segment* *Digit* 2*[*3*]*** | **3.3*V*** |
| ***HEX*2*[*4*]*** | ***PIN\_AD*30** | ***Seven* *Segment* *Digit* 2*[*4*]*** | **3.3*V*** |
| ***HEX*2*[*5*]*** | ***PIN\_AC*29** | ***Seven* *Segment* *Digit* 2*[*5*]*** | **3.3*V*** |
| ***HEX*2*[*6*]*** | ***PIN\_AC*30** | ***Seven* *Segment* *Digit* 2*[*6*]*** | **3.3*V*** |
| ***HEX*3*[*0*]*** | ***PIN\_AD*26** | ***Seven* *Segment* *Digit* 3*[*0*]*** | **3.3*V*** |
| ***HEX*3*[*1*]*** | ***PIN\_AC*27** | ***Seven* *Segment* *Digit* 3*[*1*]*** | **3.3*V*** |
| ***HEX*3*[*2*]*** | ***PIN\_AD*25** | ***Seven* *Segment* *Digit* 3*[*2*]*** | **3.3*V*** |
| ***HEX*3*[*3*]*** | ***PIN\_AC*25** | ***Seven* *Segment* *Digit* 3*[*3*]*** | **3.3*V*** |
| ***HEX*3*[*4*]*** | ***PIN\_AB*28** | ***Seven* *Segment* *Digit* 3*[*4*]*** | **3.3*V*** |
| ***HEX*3*[*5*]*** | ***PIN\_AB*25** | ***Seven* *Segment* *Digit* 3*[*5*]*** | **3.3*V*** |
| ***HEX*3*[*6*]*** | ***PIN\_AB*22** | ***Seven* *Segment* *Digit* 3*[*6*]*** | **3.3*V*** |
| ***HEX*4*[*0*]*** | ***PIN\_AA*24** | ***Seven* *Segment* *Digit* 4*[*0*]*** | **3.3*V*** |
| ***HEX*4*[*1*]*** | ***PIN\_Y*23** | ***Seven* *Segment* *Digit* 4*[*1*]*** | **3.3*V*** |
| ***HEX*4*[*2*]*** | ***PIN\_Y*24** | ***Seven* *Segment* *Digit* 4*[*2*]*** | **3.3*V*** |
| ***HEX*4*[*3*]*** | ***PIN\_W*22** | ***Seven* *Segment* *Digit* 4*[*3*]*** | **3.3*V*** |
| ***HEX*4*[*4*]*** | ***PIN\_W*24** | ***Seven* *Segment* *Digit* 4*[*4*]*** | **3.3*V*** |
| ***HEX*4*[*5*]*** | ***PIN\_V*23** | ***Seven* *Segment* *Digit* 4*[*5*]*** | **3.3*V*** |
| ***HEX*4*[*6*]*** | ***PIN\_W*25** | ***Seven* *Segment* *Digit* 4*[*6*]*** | **3.3*V*** |
| ***HEX*5*[*0*]*** | ***PIN\_V*25** | ***Seven* *Segment* *Digit* 5*[*0*]*** | **3.3*V*** |
| ***HEX*5*[*1*]*** | ***PIN\_AA*28** | ***Seven* *Segment* *Digit* 5*[*1*]*** | **3.3*V*** |
| ***HEX*5*[*2*]*** | ***PIN\_Y*27** | ***Seven* *Segment* *Digit* 5*[*2*]*** | **3.3*V*** |
| ***HEX*5*[*3*]*** | ***PIN\_AB*27** | ***Seven* *Segment* *Digit* 5*[*3*]*** | **3.3*V*** |
| ***HEX*5*[*4*]*** | ***PIN\_AB*26** | ***Seven* *Segment* *Digit* 5*[*4*]*** | **3.3*V*** |
| ***HEX*5*[*5*]*** | ***PIN\_AA*26** | ***Seven* *Segment* *Digit* 5*[*5*]*** | **3.3*V*** |
| ***HEX*5*[*6*]*** | ***PIN\_AA*25** | ***Seven* *Segment* *Digit* 5*[*6*]*** | **3.3*V*** |

П. 1. 3. Программатор.

1. Подсоедините макет к ПК и включите его.
2. Откройте программатор: ***Tools* – *Programmer*.**
3. В открывшемся окне, рис. П1. 3, проверьте наличие установки ***Hardware* *Setup* *DE*-*So*С(2 -1.6)**, или выберите эту установку через поиск ***Hardware* *Setup* – *Currently* *select* *hardware*.** Затем активируйте ***Auto* *Detect***.



Рис. П1. 3. Окно программатора.

1. В открывшемся окне ***Select* *Device*,** рис. П1. 4, выберите **5*CEMA*5**



Рис. П1. 4. Окно выбора схемы.

1. В окне программатора появится цепочка из двух схем. Поставьте курсор на вторую схему **5*CSEMA*5** и сделайте двойной клик левой кнопкой мыши. В левом боковом окне выберите ***Change* *File***.
2. В открывшемся окне ***Select* *New* *Programming* *File*** выберите ***output\_* *files*,** далее файл Вашего проекта с расширением **.*sof***.
3. В выделенной строке выберите режим программирования, отметив галочкой указанное на рис. П1. 5. окно в последнем верхнем меню.

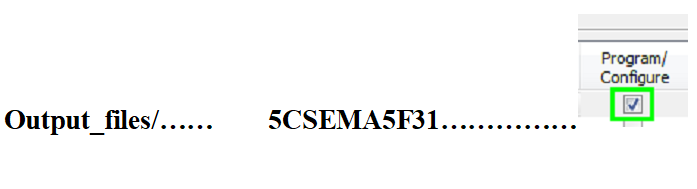


Рис. П1. 5. Выбор режима.

1. На боковом меню нажмите ***Start***.

Приложение 2.

**Задания к лабораторной работе №6.**

**Задание 1. Светофор.**

Автомат не только регулирует автомобильное движение, но и организует пешеходный переход перекрестка, где разрешен только правый поворот. Красный сигнал светофора горит в течение 6 секунд, затем подключается желтый, оба цвета горят в течение 3 секунд, причем через 2 секунды включается разрешение правого поворота. Зеленый горит также 6 секунд. Для индикации регулировки автомобильного движения задействовать 2 (3) правых индикатора (цвет + поворот). Для табло пешеходного перехода – 2 левых индикатора (цвет + счетчик времени).

**Задание 2. Стиральная машина.**

**Вариант 2.1.** Возможность выбора режима стирки.

В состоянии *S*1 автомат считывает с внешнего устройства управление, определяющее дальнейший режим работы машины. В зависимости от режима устанавливается температура, которая высвечивается на крайних левых индикаторах. Для режима стирки х/б тканей установить 400, для шерстяных - установить 300. Работа в режиме х/б: стирка 4 такта, полоскание 3 такта, отжим 2 такта; работа в режиме шерсть: стирка 3 такта, полоскание 4 такта, отжим 1 такт. Из каждого режима автомат переходит в ожидание отключения.

**Вариант 2.2.** Возможность выбора режима с проверкой веса белья.

В состоянии *S*1 автомат считывает с внешнего устройства управление, определяющее дальнейший режим работы машины. Далее, в каждом режиме производится проверка веса загруженного белья. В зависимости от режима допустимый вес может составлять: для х/б – не более 5 кг, для шерсти – не более 2 кг. При превышении веса автомат выводит машину в ожидание отключения. Работа в режиме х/б: стирка 4 такта, полоскание 3 такта, отжим 2 такта; работа в режиме шерсть: стирка 3 такта, полоскание 4 такта, отжим 1 такт. Из каждого режима автомат переходит в ожидание отключения.

**Задание 3. Работа кассового картридера с банковской картой.**

В этом задании выделяем 5 этапов:

-включение;

- проверка наличия карты «П»;

- считывание кода «С»;

- обработка кода «О»;

- получение результата «Р».

Наличие карты имитируется установкой тумблера *T*0 в «1». Считывается 4-значный код карты, предполагающий наличие на ней средств. Баланс карты задается кодом, а сумма покупки определяется константой. Если сумма покупки превышает баланс, идет возврат на этап проверки карты. В случае положительной разности, в качестве результата выводится на индикаторы сумма покупки, а на светодиоды остаток в двоичном коде.

Режим включения предусматривает проверку связи картридера с хостом. После получения результата автомат ожидает разрешения хоста на новую трансакцию - переходит в режим проверки наличия карты.

**Задание 4. Автомат задает форму разметки фигуры: при *K*=0 прямоугольная форма, при *K*=1 – круг.**

**Разметка прямоугольной фигуры.**

Движение вперед – состояние выходов - *FO*;

Движение вправо – состояние выходов *r* *I*;

Движение назад – состояние выходов *r* *E*;

Движение вправо – состояние выходов *r* *I*.

**Разметка круга** – состояние выходов – *round*.

Состояние выходов фиксируется на индикаторах.

Описание работы автомата.

Автомат имеет 8 состояний:

*S*0 – включение, удержание по *reset*. При снятии *reset* переход в *S*1.

*S*1 – ожидание установки режима. Длительность 2 такта, далее в *S*2.

*S*2 – при ***K*=1** переход в *S*7, иначе в *S*3.

*S*3 – удержание 5 тактов, код выхода автомата – 1. Переход в *S*4.

*S*4 – удержание 2 такта, код выхода автомата – 2. Переход в *S*5.

*S*5 – удержание 5 такта, код выхода автомата – 3. Переход в *S*6.

*S*6 – удержание 2 такта, код выхода автомата – 2. Переход в *S*1.

*S*7 – удержание 7 тактов, код выхода автомата – 4. Переход в *S*1.

**Задание 5.1. Последовательный вывод многозначного числа на 4 индикатора.**

Число набирается на тумблерах по 4 разряда на каждую цифру и последовательно выводится на 4 индикатора. Набранный символ записывается после установки «1» на входе разрешения записи.

1. Состояние *S*0 – начальное.
2. Состояние *S*1 – ожидание ввода. Время удержания 4 такта. Состояние выхода --.
3. Далее, автомат переходит в состояние ввода *S*2. Время удержания состояния ввода – 2 такта.
4. Автомат переходит в состояние вывода *S*3 и выводит «*\_*», если разрешение не поступит, при поступившем разрешении ввода выводится введенная цифра. Время удержания состояния вывода – 1 такт. Из состояния вывода автомат снова переходит в ожидание ввода. Последовательный вывод можно организовать с помощью регистра сдвига.

**Задание 5.2. Последовательный вывод многозначного числа на 6 индикаторов.**

Число набирается на тумблерах по 4 разряда и последовательно выводится на 6 индикаторов. Набранный символ записывается после установки «1» на входе разрешения записи.

1. Состояние *S*0 – начальное.
2. Состояние *S*1 – ожидание ввода. Время удержания 4 такта. Состояние выхода --.
3. Далее, автомат переходит в состояние ввода *S*2.. Время удержания состояния ввода – 2 такта.
4. Автомат переходит в состояние вывода *S*3 и выводит «*\_*», если разрешение не поступит, при поступившем разрешении ввода выводится введенная цифра. Время удержания состояния вывода – 1 такт. Из состояния вывода автомат снова переходит в ожидание ввода.

**Задание 6. Включение двигателя автомобиля.**

*S*0 – начальное состояние.

*S*1 – установка ключа. Этот сигнал получаем с тумблера *T*0. Ключ включает счетчик тактов. После включения счетчика время удержания состояния 2 такта. Состояние выхода – «--».

*S*2 – проверка кода. Время удержания 4 такта. Состояние выхода – «*\_\_*». Сигнал, имитирующий код, подается с тумблеров. При несовпадении переход в *S*6. При совпадении кода переход на проверку состояния топлива и масла – *S*3.

*S*3 – проверка уровня масла и наличия топлива в баке. При наличии хотя бы одного из предупреждений автомат не перейдет к включению двигателя. Если предупреждений нет, переход в *S*4.

*S*4 – включение двигателя. Состояние выхода «*F*». Время удержания 3 такта. За это время необходимо имитировать дальнейший поворот ключа с помощью тумблера *T*1, что приведет к приостановке счетчика тактов. Переход в *S*5.

*S*5 – ожидание выключения двигателя. Ключи (Т1 и Т0) приводятся в исходное положение. Переход к *S*1.

*S*6 – индикация несовпадения кода. На индикаторе горит “*A*”. Ключ (*T*0) возвращается в исходное положение. Время удержания 3 такта, переход в *S*1.

**Задание 7. Автомат для прохода в метро.**

*S*0 – начальное состояние.

*S*1 – ожидание карты. Индикация « *\_*». Приложенная карта включает счетчик тактов. Сигнал, имитирующий это действие, получаем с тумблера Т1. Через 2 такта автомат переходит в состояние *S*2 – чтение состояния карты.

*S*2 – чтение карты. Время удержания 2 такта. Состояние карты имитируется положением тумблеров, определяющих баланс карты. Состояние выхода автомата «*\_\_*». Переход в *S*3.

*S*3 – индикация состояния карты. Автомат выводит в течение 3 тактов состояние остатка на балансе при вычете тарифа. После этого автомат переходит в состояние *S*4 или *S*1(если баланс меньше тарифа).

*S*4 – вход разрешен. Тумблер Т1 выключить. Состояние выхода *GO* (турникет открыт). Время удержания 4 такта. Переход в *S*1.

**ЛИТЕРАТУРА**

1.*Угрюмов Е.П.* Цифровая схемотехника: учебное пособие для вузов [Электронный ресурс] / Е.П. Угрюмов; рец. А.И. Водяхо. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2010. – 816с.: ил, табл. – библиогр.: с. 775-780.

**Неелова Ольга Леонидовна**

**ПРОЕКТИРОВАНИЕ ЭЛЕМЕНОВ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМАХ (ПЛИС)**

**Практикум по дисциплине «Архитектура вычислительных систем»**

Редактор

План издания 2022 г, п.34

Подписано к печати .2022

Объем усл.-печ. л. Тираж экз. Заказ

Редакционно-издательский отдел СПбГУТ

193232 СПб., пр. Большевиков, 22

Отпечатано в СПбГУТ