

УТВЕРЖДАЮ
 Декан факультета

 (подпись) Страхов С. Ю.
 ФИО
 «___» _____ 20__

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ СИСТЕМ НА ПЛИС

Направление/специальность подготовки	11.05.01 Радиоэлектронные системы и комплексы
Специализация/профиль/программа подготовки	Радиолокационные системы и комплексы
Уровень высшего образования	Специалитет
Форма обучения	Очная
Факультет	И Информационных и управляющих систем
Выпускающая кафедра	И4 РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ
Кафедра-разработчик рабочей программы	И4 РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

КУРС	СЕМЕСТР	ОБЩАЯ ТРУДОЁМКОСТЬ (ЗАЧЕТНЫХ ЕДИНИЦ)	ЧАСЫ (по наличию видов занятий)									ВИД ПРОМЕЖУТОЧНОГО КОНТРОЛЯ
			ОБЩАЯ ТРУДОЁМКОСТЬ	АУДИТОРНЫЕ ЗАНЯТИЯ				САМОСТОЯТЕЛЬНАЯ РАБОТА				
				ВСЕГО	ЛЕКЦИИ	ЛАБОРАТОРНЫЙ ПРАКТИКУМ	ПРАКТИЧЕСКИЕ ЗАНЯТИЯ	ВСЕГО	КУРСОВОЙ ПРОЕКТ	КУРСОВАЯ РАБОТА	ДРУГИЕ ВИДЫ САМОСТ. РАБОТЫ	
5	9	3	108	51	17	34	0	57	0	0	57	диф. зач.

ЛИСТ СОГЛАСОВАНИЯ

**РАБОЧАЯ ПРОГРАММА СОСТАВЛЕНА В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ ФЕДЕРАЛЬНОГО
ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО СТАНДАРТА ВЫСШЕГО ОБРАЗОВАНИЯ (ФГОС ВО)**

11.05.01 Радиоэлектронные системы и комплексы

год набора группы: 2024

Программу составил:

Кафедра И4 РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ
Колачев Игорь Олегович, ассистент

Программа рассмотрена
на заседании кафедры-разработчика
рабочей программы **И4 РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ**

Заведующий кафедрой Страхов С.Ю., д.т.н., проф.

Программа рассмотрена
на заседании выпускающей кафедры

И4 РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Заведующий кафедрой Страхов С.Ю., д.т.н., проф.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ СИСТЕМ НА ПЛИС

Разделы рабочей программы

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ
2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВО
3. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ
4. ФОРМЫ КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ
5. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
6. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Приложения к рабочей программе дисциплины

- Приложение 1. Аннотация рабочей программы
- Приложение 2. Технологии и формы обучения
- Приложение 3. Фонды оценочных средств

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины является формирование следующих компетенций:

ПСК-1 — способность осуществлять анализ состояния научно-технической проблемы, определять цели и выполнять постановку задач проектирования
ПСК-4 — способность разрабатывать цифровые радиотехнические устройства на современной цифровой элементной базе с использованием современных пакетов прикладных программ

Формированию компетенций служит достижение следующих результатов образования:

ПСК-1

знания:

принципов построения цифровых устройств на ПЛИС, основных этапов их разработки;

умения:

составлять описание ПЛИС в виде набора функциональных блоков;

навыки:

функционального и структурного моделирования устройств ПЛИС.

ПСК-4

знания:

Основных этапов разработки цифровых устройств на ПЛИС;

умения:

Выбирать конструкцию ПЛИС для реализации проекта;

навыки:

Виртуального проектирования, построения программ с использованием языков и средств программирования логики.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВО

Дисциплина **ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ СИСТЕМ НА ПЛИС** является дисциплиной **части, формируемой участниками образовательных отношений блока 1**, программы подготовки по направлению *11.05.01 Радиоэлектронные системы и комплексы*.

Содержание дисциплины является логическим продолжением дисциплин: **ВВЕДЕНИЕ В СПЕЦИАЛЬНОСТЬ, МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА**.

Содержание дисциплины является основой для освоения дисциплин: **ЦИФРОВЫЕ РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ ПЕРЕДАЧИ ИЗОБРАЖЕНИЯ И ЗВУКА**.

Предварительные компетенции, сформированные у обучающегося до начала изучения дисциплины:

- ОПК-1 — Способен представить адекватную современному уровню знаний научную картину мира на основе знания основных положений, законов и методов естественных наук и математики
- ОПК-5 — Способен выполнять опытно-конструкторские работы с учетом требований нормативных документов в области радиоэлектронной техники и информационно-коммуникационных технологий
- ОПК-8 — Способен использовать современные программные и инструментальные средства компьютерного моделирования для решения различных исследовательских и профессиональных задач
- ПСК-1 — Способен осуществлять анализ состояния научно-технической проблемы, определять цели и выполнять постановку задач проектирования
- ПСК-2 — Способен разрабатывать структурные и функциональные схемы радиоэлектронных систем и комплексов, а также принципиальные схемы радиоэлектронных устройств с применением современных САПР и пакетов прикладных программ
- ПСК-3 — Способен осуществлять проектирование конструкций электронных средств с применением современных САПР и пакетов прикладных программ
- ПСК-4 — Способен разрабатывать цифровые радиотехнические устройства на современной цифровой элементной базе с использованием современных пакетов прикладных программ
- УК-6 — Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки и образования в течение всей жизни

3. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Общая трудоемкость дисциплины составляет 3 з.е., 108 ч.

3.1. Содержание (дидактика) дисциплины

КУРС	СЕМЕСТР	Наименование разделов и дидактических единиц	ВСЕГО	Аудиторные занятия в контактной форме			Самостоятельная работа студентов	Формируемая компетенция, %	
				ВСЕГО	Лекции	Лабораторный практикум		ПСК-1	ПСК-4
5	9	Раздел 1. Введение. Основы архитектуры ПЛИС. История ПЛИС. Основные блоки ПЛИС. Основные типы архитектур ПЛИС. Ведущие производители ПЛИС. Обзор актуальных архитектур ПЛИС. Обзор систем автоматизированного проектирования. Обзор языков описания аппаратуры (HDL). Этапы компиляции проектов под ПЛИС. Этапы симуляции проектов.	11	2	2	0	9	16	16
5	9	Раздел 2. Основы языка описания аппаратуры Verilog. История языка Verilog. Прimitives. Модули. Порты. Синтезируемые и несинтезируемые конструкции. Типы данных. Непрерывное присваивание. Операторы. Представление чисел. Процедурные блоки. Блокирующее присваивание. Неблокирующее присваивание. Конструкции условного выбора. Использование IP-ядер. Тестирующие модули (тестбенчи). Конструкции языка, используемые для симуляции.	18	9	3	6	9	16	16
5	9	Раздел 3. Комбинационные устройства. Шифраторы. Дешифраторы. Преобразователи кодов. Селекторы Мультиплексоры. Демультимплексоры. Сумматоры. Компараторы. Устройства сдвига. Арифметико-логические устройства (АЛУ). Параметризованные модули. Конструкции генерации.	19	10	3	7	9	17	17
5	9	Раздел 4. Последовательностные устройства. Триггеры. Регистры. Сдвиговые регистры. Счётчики. Элементы памяти. Регистровый файл. Постоянные запоминающие устройства (ПЗУ). Оперативные запоминающие устройства. Стек. Конвейерная обработка.	20	10	3	7	10	17	17
5	9	Раздел 5. Конечные автоматы. Основные идеи конечных автоматов. Конечный автомат Мура. Конечный автомат Мили. Графы и таблицы переходов конечных автоматов. Таблицы выходов конечных автоматов. Кодирование состояний. Этапы проектирования конечных автоматов. Методы описания конечных автоматов. Реализация протоколов UART, SPI и I2C с помощью конечных автоматов. Реализация алгоритма CORDIC с помощью конечных автоматов.	21	11	4	7	10	17	17
5	9	Раздел 6. Использование процессорного модуля. Программные и аппаратные процессорные модули. Настройка шин адреса и данных в процессорной системе. Настройка такого сигнала и сигнала сброса. Настройка адресного пространства и вектора прерываний. Использование блоков параллельного ввода-вывода (PIO). Использование пользовательских модулей. Программирование процессорного модуля на языке C.	19	9	2	7	10	17	17
Всего за 9 семестр			108	51	17	34	57	100	100
Всего по дисциплине			108	51	17	34	57	100	100

3.2. Лабораторный практикум

№ п/п	Номер и наименование раздела дисциплины	Тема лабораторного практикума	Объем, ауд. часов
1	Раздел 2. Основы языка описания аппаратуры Verilog.	Знакомство с САПР Quartus. Проектирование и моделирование простейшего комбинационного устройства.	6
2	Раздел 3. Комбинационные устройства.	Разработка параметризованного комбинационного устройства.	7
3	Раздел 4. Последовательностные устройства.	Проектирование простейшего цифрового генератора сигналов.	7
4	Раздел 5. Конечные автоматы.	Проектирование цифрового устройства на основе конечного автомата.	7
5	Раздел 6. Использование процессорного модуля.	Проектирование устройства управления на основе процессорного модуля.	7
Всего за 9 семестр			34

3.3. Самостоятельная работа студента (СРС)

№ п/п	Номер и наименование раздела дисциплины	Содержание учебного задания	Объем, часов
1	Раздел 1. Введение.	Изучение особенностей дисциплины, знакомство с	9

	Основы архитектуры ПЛИС.	рекомендуемой литературой.	
2	Раздел 2. Основы языка описания аппаратуры Verilog.	Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	9
3	Раздел 3. Комбинационные устройства.	Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	9
4	Раздел 4. Последовательностные устройства.	Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	10
5	Раздел 5. Конечные автоматы.	Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	10
6	Раздел 6. Использование процессорного модуля.	Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	10
Всего за 9 семестр			57

4. ФОРМЫ КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

СЕМЕСТР	НЕДЕЛИ СЕМЕСТРА																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
9				ЛР		ДР	ЛР		ЛР	ДР		ЛР		ЛР		ДР	Вопр.Диф.Зач, диф. зач.

Условные обозначения:

- ДР – диагностическая работа;
- Вопр.Диф.Зач – вопросы к дифференцированному зачету;
- ЛР – лабораторная работа;
- диф. зач. – дифференцированный зачет.

Текущий контроль успеваемости студентов проводится в дискретные временные интервалы в следующих формах:

- диагностическая работа;
- вопросы к дифференцированному зачету;
- лабораторная работа.

Промежуточная аттестация проводится в формах:

- дифференцированный зачет.

5. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

5.1. Основная литература по дисциплине:

1. А. К. Нарышкин. . Цифровые устройства и микропроцессоры. М.: Академия, 2008, 200 экз.
2. А. М. Сажнев. . Цифровые устройства и микропроцессоры. Москва: Юрайт, 2020, эл. рес.
3. А. Х. Мурсаев, О. И. Буренева. . Практикум по проектированию на языках VerilogHDL и SystemVerilog. Санкт-Петербург: Лань, 2022, эл. рес.
4. Д. М. Харрис, С. Л. Харрис. . Цифровая схемотехника и архитектура компьютера. Waltham: Morgan Kaufman, 2013, эл. рес.
5. К. Максфилд. . Проектирование на ПЛИС. Архитектура, средства и методы. Курс молодого бойца. М.: ДОДЭКА-XXI, 2007, эл. рес.
6. О. Н. Музыченко. Методы синтеза конечных автоматов. СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2012, 73 экз.

5.2. Дополнительная литература по дисциплине:

не требуется.

5.3. Периодические издания:

не требуются.

5.4. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины, электронные библиотечные системы:

1. <http://library.voenmeh.ru> — Фундаментальная библиотека БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова;
2. <http://e.lanbook.com> — ЭБС Лань;
3. <http://urait.ru> — Образовательная платформа «Юрайт». Для вузов и ссузов.;
4. <https://marsohod.org/> — FPGA блог: опыт, отладка, программирование на Verilog;
5. <https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=836> — Terasic - SoC Platform - Cyclone - DE1-SoC Board;
6. <https://e.lanbook.com/book/322511> — ЭБС Лань;
7. <https://kit-e.ru/kratkij-kurs/> — Краткий курс HDL. Язык Verilog.

Современные профессиональные базы данных:

1. <https://rusneb.ru> – Национальная электронная библиотека (НЭБ);
2. <https://cyberleninka.ru/> - Научная электронная библиотека «Киберленинка»;
<http://www.rfbr.ru/rffi/ru/library> - Полнотекстовая электронная библиотека Российского фонда фундаментальных исследований.

Информационные справочные системы:

1. Техэксперт – Информационный портал технического регулирования: Нормы, правила, стандарты РФ;
2. http://library.voenmeh.ru/jirbis2/index.php?option=com_irbis&view=irbis&Itemid=457 - БД ГОСТов собственной генерации БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова;
3. <http://www.consultant.ru/>- КонсультантПлюс- информационный портал правовой информации.

5.5. Программное обеспечение:

не требуется.

5.6. Информационные технологии:

взаимодействие с обучающимися посредством ЭИОС Moodle БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова.

6. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

6.1. Лекционные занятия:

специализированные требования по оборудованию отсутствуют; аудитория с посадочными местами по количеству студентов; доска.

6.2. Лабораторные занятия:

1. Проектор.

6.3. Прочее:

1. рабочее место преподавателя, оснащенное компьютером с доступом в Интернет;
2. рабочие места студентов, оснащенные компьютерами с доступом в Интернет, предназначенные для работы в электронной образовательной среде.

Аннотация рабочей программы

Дисциплина **ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ СИСТЕМ НА ПЛИС** является дисциплиной **части, формируемой участниками образовательных отношений блока 1**, программы подготовки по направлению *11.05.01 Радиоэлектронные системы и комплексы*. Дисциплина реализуется на факультете *И Информационных и управляющих систем БГТУ "ВОЕНМЕХ" им. Д.Ф. Устинова* кафедрой **И4 РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ**.

Дисциплина нацелена на формирование *компетенций*:

ПСК-1 способность осуществлять анализ состояния научно-технической проблемы, определять цели и выполнять постановку задач проектирования;

ПСК-4 способность разрабатывать цифровые радиотехнические устройства на современной цифровой элементной базе с использованием современных пакетов прикладных программ.

Содержание дисциплины охватывает круг вопросов, связанных с проектированием цифровых систем на современных программируемых логических интегральных схемах с использованием языка описания аппаратуры Verilog.

Программой дисциплины предусмотрены следующие **виды контроля**:

Текущий контроль успеваемости студентов проводится в дискретные временные интервалы в следующих формах:

- диагностическая работа;
- вопросы к дифференцированному зачету;
- лабораторная работа.

Промежуточная аттестация проводится в формах:

- дифференцированный зачет.

Общая трудоемкость освоения дисциплины составляет **3 з.е., 108 ч**. Программой дисциплины предусмотрены лекционные занятия (**17 ч.**), лабораторный практикум (**34 ч.**), самостоятельная работа студента (**57 ч.**).

ТЕХНОЛОГИИ И ФОРМЫ ОБУЧЕНИЯ

Рекомендации по освоению дисциплины для студента

Трудоемкость освоения дисциплины составляет 108 ч., из них 51 ч. аудиторных занятий, и 57 ч., отведенных на самостоятельную работу студента.

Рекомендации по распределению учебного времени по видам самостоятельной работы и разделам дисциплины приведены в таблице.

Контроль освоения дисциплины производится в соответствии с Положением о текущем, рубежном контроле успеваемости и промежуточной аттестации обучающихся.

Формы контроля и критерии оценивания приведены в приложении 3 к Рабочей программе.

Наименование работы	Рекомендуемая литература	Трудоемкость, час.
Раздел 1. Введение. Основы архитектуры ПЛИС.		
Изучение особенностей дисциплины, знакомство с рекомендуемой литературой.	К. Максфилд. . Проектирование на ПЛИС. Архитектура, средства и методы. Курс молодого бойца: М.: ДОДЭКА-XXI, 2007 (2-6)	9
Итого по разделу 1		9
Раздел 2. Основы языка описания аппаратуры Verilog.		
Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	А. Х. Мурсаев, О. И. Буренева. . Практикум по проектированию на языках VerilogHDL и SystemVerilog: Санкт-Петербург: Лань, 2022 (1) Д. М. Харрис, С. Л. Харрис. . Цифровая схемотехника и архитектура компьютера: Waltham: Morgan Kaufman, 2013 (4-5) К. Максфилд. . Проектирование на ПЛИС. Архитектура, средства и методы. Курс молодого бойца: М.: ДОДЭКА-XXI, 2007 (9, 16)	9
Итого по разделу 2		9
Раздел 3. Комбинационные устройства.		
Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	Д. М. Харрис, С. Л. Харрис. . Цифровая схемотехника и архитектура компьютера: Waltham: Morgan Kaufman, 2013 (2, 4-5) А. М. Сажнев. . Цифровые устройства и микропроцессоры: Москва: Юрайт, 2020 (1-3) А. К. Нарышкин. . Цифровые устройства и микропроцессоры: М.: Академия, 2008 (6-12) А. Х. Мурсаев, О. И. Буренева. . Практикум по проектированию на языках VerilogHDL и SystemVerilog: Санкт-Петербург: Лань, 2022 (2)	9
Итого по разделу 3		9
Раздел 4. Последовательностные устройства.		
Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из	Д. М. Харрис, С. Л. Харрис. . Цифровая схемотехника и	10

библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	архитектура компьютера: Waltham: Morgan Kaufman, 2013 (5) А. М. Сажнев. . Цифровые устройства и микропроцессоры: Москва: Юрайт, 2020 (4) А. К. Нарышкин. . Цифровые устройства и микропроцессоры: М.: Академия, 2008 (13-16) А. Х. Мурсаев, О. И. Буренева. . Практикум по проектированию на языках VerilogHDL и SystemVerilog: Санкт-Петербург: Лань, 2022 (2)	
Итого по разделу 4		10
Раздел 5. Конечные автоматы.		
Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	А. Х. Мурсаев, О. И. Буренева. . Практикум по проектированию на языках VerilogHDL и SystemVerilog: Санкт-Петербург: Лань, 2022 (4) Д. М. Харрис, С. Л. Харрис. . Цифровая схемотехника и архитектура компьютера: Waltham: Morgan Kaufman, 2013 (3) О. Н. Музыченко. Методы синтеза конечных автоматов: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2012 (1-4)	10
Итого по разделу 5		10
Раздел 6. Использование процессорного модуля.		
Изучение рекомендованной литературы. Изучение примеров проектирования систем на ПЛИС из библиотеки Quartus. Подготовка к выполнению лабораторной работы. Подготовка отчёта по лабораторной работе.	К. Максфилд. . Проектирование на ПЛИС. Архитектура, средства и методы. Курс молодого бойца: М.: ДОДЭКА-XXI, 2007 (13) А. Х. Мурсаев, О. И. Буренева. . Практикум по проектированию на языках VerilogHDL и SystemVerilog: Санкт-Петербург: Лань, 2022 (7)	10
Итого по разделу 6		10

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

Фонд оценочных средств, позволяющие оценить результаты обучения по данной дисциплине, включают в себя:

- диагностическая работа
- вопросы к дифференцированному зачету;
- лабораторная работа;
- дифференцированный зачет.

Критерии оценивания

Диагностическая работа

Диагностическая работа проводится в форме теста в ЭИОС Moodle:

- при правильном ответе менее чем на 60% вопросов - не аттестация;
- при правильном ответе на 60% вопросов и более - аттестация.

Вопросы к дифференцированному зачету

Перечень вопросов опубликован в ЭИОС Moodle на странице курса.

Лабораторная работа

После выполнения лабораторной работы, студент готовит отчет, сдает его преподавателю, а далее происходит собеседование, направленное на выявление степени усвоения материала.

Дифференцированный зачет

Дифференцированный зачет сдается при условии полного выполнения графика контрольных мероприятий. На дифф. зачете студенту выдается тест из 10 вопросов. При правильном ответе на 6 вопросов студент получает "удовлетворительно", при правильном ответе на 7-9 вопросов - "хорошо", при правильном ответе на 10 вопросов - "отлично".

Паспорт фонда оценочных средств

КУРС	СЕМЕСТР	Наименование разделов и дидактических единиц	ВСЕГО	Аудиторные занятия в контактной форме			Самостоятельная работа студентов	Формируемая компетенция, %		НАИМЕНОВАНИЕ ОЦЕНОЧНОГО СРЕДСТВА
				ВСЕГО	Лекции	Лабораторный практикум		ПСК-1	ПСК-4	
5	9	Раздел 1. Введение. Основы архитектуры ПЛИС.	11	2	2	0	9	16	16	Вопросы к дифференцированному зачету
5	9	Раздел 2. Основы языка описания аппаратуры Verilog.	18	9	3	6	9	16	16	Лабораторная работа
5	9	Раздел 3. Комбинационные устройства.	19	10	3	7	9	17	17	Лабораторная работа
5	9	Раздел 4. Последовательностные устройства.	20	10	3	7	10	17	17	Лабораторная работа
5	9	Раздел 5. Конечные автоматы.	21	11	4	7	10	17	17	Лабораторная работа
5	9	Раздел 6. Использование процессорного модуля.	19	9	2	7	10	17	17	Лабораторная работа
Всего за 9 семестр			108	51	17	34	57	100	100	
Всего по дисциплине			108	51	17	34	57	100	100	

Критерии оценивания

ПСК-1

Вопросы открытого типа:

- № 1 Какое ключевое слово языка Verilog отвечает за открывающуюся процедурную скобку?
- № 2 Какое ключевое слово языка Verilog отвечает за закрывающуюся процедурную скобку?
- № 3 Какое ключевое слово языка Verilog отвечает за объявление блока выбора?
- № 4 Какое ключевое слово языка Verilog отвечает за объявление значения по умолчанию в блоке case?
- № 5 Какое ключевое слово языка Verilog отвечает за объявление модуля?
- № 6 Какое ключевое слово языка Verilog отвечает за окончание объявления модуля?
- № 7 Какое ключевое слово языка Verilog отвечает за условный оператор?
- № 8 Какое ключевое слово языка Verilog отвечает за объявление входного порта?
- № 9 Какое ключевое слово языка Verilog отвечает за объявление двунаправленного порта?
- № 10 Какое ключевое слово языка Verilog отвечает за объявление выходного порта?
- № 11 Какое ключевое слово языка Verilog отвечает за объявление блока, запускающегося один раз при начале моделирования?
- № 12 Какое ключевое слово языка Verilog отвечает за объявление блока, в котором допускается использовать блокирующее и неблокирующее присваивание?
- № 13 Какое ключевое слово языка Verilog отвечает за объявление непрерывного присваивания?
- № 14 Какое ключевое слово языка Verilog отвечает за объявление непрерывного присваивания?
- № 15 Какое ключевое слово языка Verilog отвечает за объявление регистра?
- № 16 Какое ключевое слово языка Verilog отвечает за объявление проводника?
- № 17 Какое ключевое слово языка Verilog обозначает фронт сигнала?
- № 18 Какое ключевое слово языка Verilog обозначает спад сигнала?
- № 19 С помощью какого оператора языка Verilog осуществляется неблокирующее присваивание?
- № 20 С помощью какого оператора языка Verilog осуществляется блокирующее присваивание?
- № 21 С помощью какого оператора языка Verilog осуществляется инверсия сигнала?
- № 22 Каким символом в языке Verilog обозначается неизвестное состояние?
- № 23 Каким символом в языке Verilog обозначается высокоимпедансное состояние?

Вопросы закрытого типа:

- № 1 На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?

```
`timescale 1ns/1ns
```

```
module Q1;
```

```
    reg [1:0] data;
```

```
    initial begin
```

```
        data = 2'b00;
```



```
        #10;
```

```
        data = 2'b01;
```

```
        #10;
```

```
        data = 2'b10;
```

```
        #10;
```

	<pre> data = 2'b11; #10; end initial \$dumpvars; endmodule </pre>
№ 2	<div>  <p>На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?</p> <pre> `timescale 1ns/1ns module Q1; reg [1:0] data; initial begin data = 2'b00; #20; data = 2'b01; #20; data = 2'b10; #20; data = 2'b11; #20; end initial \$dumpvars; endmodule </pre> </div>
№ 3	<div>  <p>На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?</p> <pre> `timescale 1ns/1ns module Q1; reg [1:0] data; initial begin data = 2'b11; #10; </pre> </div>


```

data = 2'b00;

#10;

data = 2'b11;

#10;

data = 2'b00;

#10;

end

initial

    $dumpvars;

endmodule

```

№ 4

На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?

```

`timescale 1ns/1ns

module Q1;

    reg [1:0] data;

    initial begin

        data = 2'b01;

        #10;

        data = 2'b10;

        #10;

        data = 2'b01;

        #10;

        data = 2'b10;

        #10;

    end

    initial

        $dumpvars;

endmodule

```

№ 5

На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?

```

`timescale 1ns/1ns

```

```

module Q2;

    reg [2:0] data;

    initial begin

        data = 3'b101;

        #10;

        data = 3'b100;

        #10;

        data = 3'b110;

        #10;

        data = 3'b010;

        #10;

    end

    initial

        $dumpvars;

endmodule

```

<figure class="image"></figure>

№ 6

На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?

```

`timescale 1ns/1ns

module Q2;

    reg [2:0] data;

    initial begin

        data = 3'b000;

        #10;

        data = 3'b001;

        #10;

        data = 3'b010;

        #10;

        data = 3'b100;



        #10;



    end

    initial

        $dumpvars;

```

№ 7	<div data-bbox="475 73 1348 212" data-label="Text">  </div> <p>На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?</p> <pre> timescale 1ns/1ns module Q2; reg [2:0] data; initial begin data = 3'b000; #10; data = 3'b001; #20; data = 3'b010; #20; data = 3'b100; #10; end initial \$dumpvars; endmodule </pre>
№ 8	<div data-bbox="475 1355 1348 1422" data-label="Text">  </div> <p>На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?</p> <pre> timescale 1ns/1ns module Q2; reg [2:0] data; initial begin data = 3'b101; #20; data = 3'b100; #20; data = 3'b110; #10; end initial \$dumpvars; endmodule </pre>

	<pre> data = 3'b010; #10; end initial \$dumpvars; endmodule </pre>
№ 9	<div>  <p>На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?</p> </div> <pre> `timescale 1ns/1ns module Q3; reg [3:0] data; initial begin data = 4'b1000; #10; data = 4'b0100; #10; data = 4'b0010; #10; data = 4'b0001; #10; end initial \$dumpvars; endmodule </pre>
№ 10	<div>  <p>На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?</p> </div> <pre> `timescale 1ns/1ns module Q3; reg [3:0] data; initial begin data = 4'b0001; #10; </pre>

№ 11

```
data = 4'b0010;

#10;

data = 4'b0100;

#10;

data = 4'b1000;

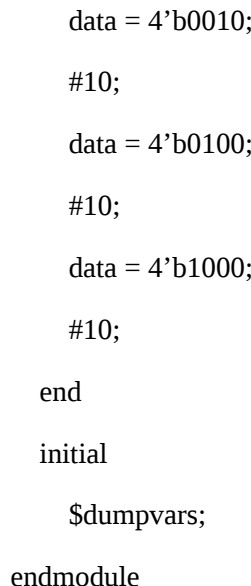
#10;

end

initial

    $dumpvars;

endmodule
```



На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?

```
`timescale 1ns/1ns

module Q3;

    reg [3:0] data;

    initial begin

        data = 4'b1000;

        #10;

        data = 4'b0100;

        #20;

        data = 4'b0010;

        #10;

        data = 4'b0001;

        #20;

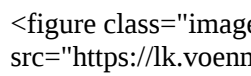
    end

    initial

        $dumpvars;

endmodule
```

№ 12



На какой из приведённых временных диаграмм приведен результат моделирования следующего кода?

```
`timescale 1ns/1ns

module Q3;
```

```

reg [3:0] data;

initial begin

    data = 4'b0001;

    #20;

    data = 4'b0010;

    #10;

    data = 4'b0100;

    #10;

    data = 4'b1000;

    #20;

end

initial

    $dumpvars;

endmodule

```



№ 13

На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```

module func(

input [1:0] a, b,

input [1:0] op,

output reg c

);

always @(*)

case (op)

2'b00: c = a == b;

2'b01: c = a != b;

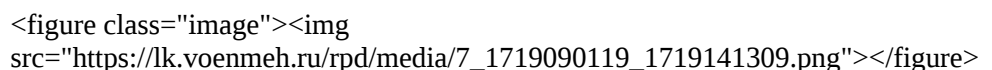
2'b10: c = a < b;

2'b11: c = a > b;

endcase

endmodule

```



№ 14

На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```

module func(

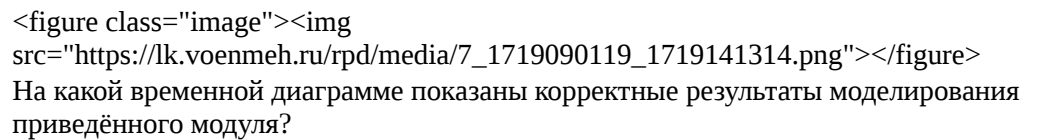
```

```

input [1:0] a, b,
input [1:0] op,
output reg c
);
always @(*)
case (op)
2'b00: c = a == b;
2'b01: c = a != b;
2'b10: c = a <= b;
2'b11: c = a >= b;
endcase
endmodule

```

№ 15

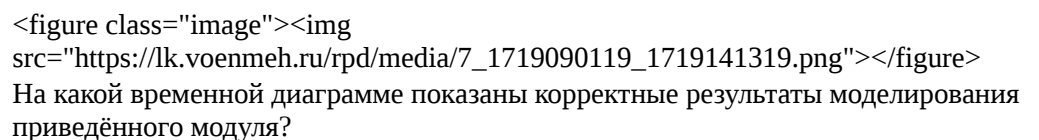
 На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```

module func(
input [1:0] a, b,
input [1:0] op,
output reg c
);
always @(*)
case (op)
2'b00: c = a == b;
2'b01: c = a != b;
2'b10: c = a == b;
2'b11: c = a != b;
endcase
endmodule

```

№ 16

 На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```

module func(
input [1:0] a, b,
input [1:0] op,

```

```

output reg c

);

always @(*)

case (op)

  2'b00: c = a <= b;

  2'b01: c = a != b;

  2'b10: c = a >= b;

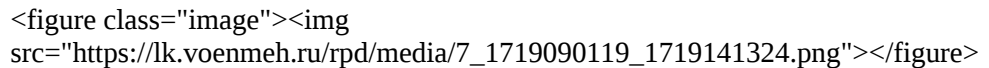
  2'b11: c = a != b;

endcase

endmodule

```

№ 17


 На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```

module func(

input [1:0] a, b,

input [1:0] op,

output reg [1:0] c

);

always @(*)

case (op)

  2'b00: c = a & b;

  2'b01: c = a | b;

  2'b10: c = a ^ b;

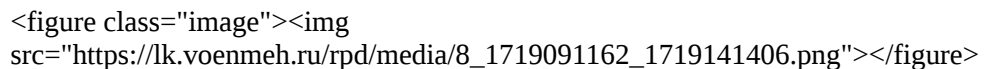
  2'b11: c = ~a;

endcase

endmodule

```

№ 18


 На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```

module func(

input [1:0] a, b,

input [1:0] op,

output reg [1:0] c

);

```



```

always @(*)
case (op)
2'b00: c = a & b;
2'b01: c = a | b;
2'b10: c = a ^ b;
2'b11: c = ~b;
endcase
endmodule

```

№ 19

<figure class="image"></figure>
На какой временной диаграмме показаны корректные результаты моделирования
приведённого модуля?

```

module func(
input [1:0] a, b,
input [1:0] op,
output reg [1:0] c
);
always @(*)
case (op)
2'b00: c = ~(a & b);
2'b01: c = ~(a | b);
2'b10: c = a ^ b;
default: c = ~a;
endcase
endmodule

```

№ 20

<figure class="image"></figure>
На какой временной диаграмме показаны корректные результаты моделирования
приведённого модуля?

```

module func(
input [1:0] a, b,
input [1:0] op,
output reg [1:0] c
);
always @(*)
case (op)

```

```
2'b00: c = ~(a & b);
```

```
2'b01: c = ~(a | b);
```

```
2'b10: c = a ^ b;
```

```
default: c = ~b;
```

```
endcase
```

```
endmodule
```

```
<figure class="image"></figure>
```

№ 21

На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```
module func(
```

```
input [7:0] a, b,
```

```
input [1:0] op,
```

```
output reg [7:0] c
```

```
);
```

```
always @(*)
```

```
case (op)
```

```
2'b00: c = a + b;
```

```
2'b01: c = a - b;
```

```
default: c = 7'd39;
```

```
endcase
```

```
endmodule
```

```
<figure class="image"></figure>
```

№ 22

На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```
module func(
```

```
input [7:0] a, b,
```

```
input [1:0] op,
```

```
output reg [7:0] c
```

```
);
```

```
always @(*)
```

```
case (op)
```

```
2'b00: c = a * b;
```

```
2'b01: c = a / b;
```

```
2'b10: c = a % b;
```

№ 23

```
default: c = 7'd39;
```

```
endcase
```

```
endmodule
```

```
<figure class="image"></figure>
```

На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```
module func(  
input [7:0] a, b,  
input [1:0] op,  
output reg [7:0] c  
);
```

```
always @(*)
```

```
case (op)
```

```
2'b00: c = a * b;
```

```
2'b01: c = a % b;
```

```
2'b10: c = a + b;
```

```
2'b10: c = a - b;
```

```
endcase
```

```
endmodule
```

№ 24

```
<figure class="image"></figure>
```

На какой временной диаграмме показаны корректные результаты моделирования приведённого модуля?

```
module func(  
input [7:0] a, b,  
input [1:0] op,  
output reg [7:0] c  
);
```

```
always @(*)
```

```
case (op)
```

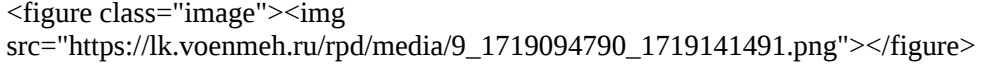
```
2'b00: c = 2 * a;
```

```
2'b01: c = 3 * a;
```

```
2'b10: c = 2 * b;
```

```
2'b11: c = 3 * b;
```

```
endcase
```

- endmodule
- № 25  Какой логической функции эквивалентен приведённый примитив?
- ```

primitive block(
 output q,
 input a, b, c);
table
 1 1 1 : 1;
 ? ? ? : 0;
endtable
endprimitive

```
- А. И
- Б. И-НЕ
- В. ИЛИ
- Г. ИЛИ-НЕ
- № 26 Какой логической функции эквивалентен приведённый примитив?
- ```

primitive block(
    output q,
    input a, b, c);
table
    1 1 1 : 0;
    0 ? ? : 1;
    ? 0 ? : 1;
    ? ? 0 : 1;
endtable
endprimitive

```
- А. И
- Б. И-НЕ
- В. ИЛИ
- Г. ИЛИ-НЕ
- № 27 Какой логической функции эквивалентен приведённый примитив?
- ```

primitive block(
 output q,
 input a, b, c);

```

table

0 0 0 : 0;

1 ? ? : 1;

? 1 ? : 1;

? ? 1 : 1;

endtable

endprimitive

А. И

Б. И-НЕ

В. ИЛИ

Г. ИЛИ-НЕ

№ 28

Какой логической функции эквивалентен приведённый примитив?

primitive block(

output q,

input a, b, c);

table

0 0 0 : 1;

1 ? ? : 0;

? 1 ? : 0;

? ? 1 : 0;

endtable

endprimitive

А. И

Б. И-НЕ

В. ИЛИ

Г. ИЛИ-НЕ

№ 29

Какому блоку эквивалентен приведённый примитив?

primitive block(

output q,

input a, b, c);

table

1 1 1 : 1;

1 0 0 : 1;

0 1 0 : 1;

0 0 1 : 1;

??? : 0;

endtable

endprimitive

А. Исключающие ИЛИ

Б. Исключающие ИЛИ-НЕ

В. Мажоритарный оператор

Г. Мультиплексор

№ 30

Какому блоку эквивалентен приведённый примитив?

primitive block(

output q,

input a, b, c);

table

1 1 1 : 0;

1 0 0 : 0;

0 1 0 : 0;

0 0 1 : 0;

??? : 1;

endtable

endprimitive

А. Исключающие ИЛИ

Б. Исключающие ИЛИ-НЕ

В. Мажоритарный оператор

Г. Мультиплексор

№ 31

Какому блоку эквивалентен приведённый примитив?

primitive block(

output q,

input a, b, c);

table

1 1 ? : 1;

1 ? 1 : 1;

? 1 1 : 1;

??? : 0;

endtable

endprimitive

А. Исключающие ИЛИ

- Б. Исключающие ИЛИ-НЕ
- В. Мажоритарный оператор
- Г. Мультиплексор
- № 32 Какому блоку эквивалентен приведённый примитив?
- ```
primitive block(
    output q,
    input a, b, c);
    table
        0 0 ? : 0;
        0 1 ? : 1;
        1 ? 0 : 0;
        1 ? 1 : 1;
    endtable
endprimitive
```
- А. Исключающие ИЛИ
- Б. Исключающие ИЛИ-НЕ
- В. Мажоритарный оператор
- Г. Мультиплексор
- № 33 Какой встроенный примитив языка Verilog эквивалентен логической функции исключающие ИЛИ?
- А. xor
- Б. nor
- В. xnor
- Г. and
- № 34 Какой встроенный примитив языка Verilog эквивалентен логической функции исключающие ИЛИ-НЕ?
- А. xor
- Б. nor
- В. xnor
- Г. and

ПСК-4

Вопросы открытого типа:

- № 1 Запишите число 25 (десятичная система) в десятичной системе счисления на языке Verilog, используя 8 разрядов.
- № 2 Запишите число 101 (двоичная система) в десятичной системе счисления на языке Verilog, используя 4 разряда.
- № 3 Запишите число 101 (восьмеричная система) в десятичной системе счисления на языке Verilog, используя 12 разрядов.
- № 4 1. Запишите число 2A (шестнадцатеричная система) в десятичной системе счисления на языке Verilog, используя 16 разрядов.
- № 5 Запишите число 17 (десятичная система) в двоичной системе счисления на языке

- Verilog, используя 5 разрядов.
- № 6 Запишите число 1110 (двоичная система) в двоичной системе счисления на языке Verilog, используя 4 разряда
- № 7 Запишите число 11 (восьмеричная система) в двоичной системе счисления на языке Verilog, используя 10 разрядов
- № 8 Запишите число 2В (шестнадцатеричная система) в двоичной системе счисления на языке Verilog, используя 16 разрядов.
- № 9 Запишите число 21 (десятичная система) в восьмеричной системе счисления на языке Verilog, используя 6 разрядов.
- № 10 Запишите число 1000 (двоичная система) в восьмеричной системе счисления на языке Verilog, используя 4 разряда.
- № 11 Запишите число 123 (восьмеричная система) в восьмеричной системе счисления на языке Verilog, используя 8 разрядов.
- № 12 Запишите число E (шестнадцатеричная система) в восьмеричной системе счисления на языке Verilog, используя 5 разрядов.
- № 13 Запишите число 33 (десятичная система) в шестнадцатеричной системе счисления на языке Verilog, используя 8 разрядов.
- № 14 Запишите число 110111 (двоичная система) в шестнадцатеричной системе счисления на языке Verilog, используя 9 разрядов.
- № 15 Запишите число 27 (восьмеричная система) в шестнадцатеричной системе счисления на языке Verilog, используя 6 разрядов.
- № 16 Запишите число AE (шестнадцатеричная система) в шестнадцатеричной системе счисления на языке Verilog, используя 9 разрядов.
- № 17 Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \wedge x_1 \vee x_2 \wedge x_3 \vee x_4.$$

- ```

module testquestion(
 input [4:0] x,
 output y
);
//пользовательский код
endmodule

```
- № 18 Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = \overline{x_0} \wedge x_1 \wedge x_2 \vee \overline{x_3 \wedge x_4}$$

- ```

module testquestion(
    input [4:0] x,
    output y
);
//пользовательский код
endmodule

```
- № 19 Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = \overline{x_0} \wedge (x_1 \wedge \overline{x_2} \vee x_3 \wedge x_4)$$

```
module testquestion(
```

```
    input [4:0] x,
```

```
    output y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 20

Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \wedge \overline{(x_1 \wedge \overline{x_2} \vee x_3 \wedge x_4)}$$

```
module testquestion(
```

```
    input [4:0] x,
```

```
    output y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 21

Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \vee \overline{(x_1 \wedge \overline{x_2} \wedge x_3 \wedge x_4)}$$

```
module testquestion(
```

```
    input [4:0] x,
```

```
    output y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 22

Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \wedge (x_1 \vee x_2 \vee x_3) \vee \overline{x_0} \wedge x_4$$

```
module testquestion(
```

```

        input [4:0] x,
        output y
    );

    //пользовательский код

endmodule

```

- № 23 Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \oplus x_2 \vee \overline{(x_1 \wedge x_3 \wedge x_4)}$$

```

module testquestion(
    input [4:0] x,
    output y
);

    //пользовательский код

endmodule

```

- № 24 Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \oplus \overline{(x_1 \wedge x_2 \wedge x_3 \wedge x_4)}$$

```

module testquestion(
    input [4:0] x,
    output y
);

    //пользовательский код

endmodule

```

- № 25 Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = x_0 \oplus x_4 \vee \overline{(x_2 \oplus x_3)} \wedge x_1$$

```

module testquestion(
    input [4:0] x,
    output y
);

    //пользовательский код

endmodule

```

№ 26

Используя приведённый шаблон модуля, реализовать следующую логическую функцию с помощью непрерывного присваивания:

$$y = \overline{(x_0 \oplus x_4 \vee x_2 \oplus x_3)} \vee x_1$$

```
module testquestion(
```

```
    input [4:0] x,
```

```
    output y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 27

Используя приведённый шаблон модуля, реализовать с помощью тернарного оператора двухвходовой однобитный мультиплексор. Когда адресный вход sel=0, к выходу должен быть подключён вход d0, в противном случае – d1.

```
module testquestion(
```

```
    input d0, d1, sel,
```

```
    output y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 28

Используя приведённый шаблон модуля, реализовать с помощью оператора выбора if/else двухвходовой однобитный мультиплексор. Когда адресный вход sel=0, к выходу должен быть подключён вход d0, в противном случае – d1.

```
module testquestion(
```

```
    input d0, d1, sel,
```

```
    output reg y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 29

Используя приведённый шаблон модуля, реализовать с помощью оператора множественного выбора двухвходовой однобитный мультиплексор. Когда адресный вход sel=0, к выходу должен быть подключён вход d0, в противном случае – d1.

```
module testquestion(
```

```
    input d0, d1, sel,
```

```
    output reg y
```

```
);
```

```
//пользовательский код
```

```
endmodule
```

№ 30

Используя приведённый шаблон модуля, реализовать с помощью оператора

множественного выбора трёхходовой однобитный мультиплексор. Когда адресный вход sel=0, к выходу должен быть подключён вход d0, sel=1 – d1 и так далее. Значение по умолчанию должно быть равно 0.

```
module testquestion(
    input d0, d1, d2,
    input [1:0] sel,
    output reg y
);
```

//пользовательский код

endmodule

№ 31

Используя приведённый шаблон модуля, реализовать с помощью оператора множественного выбора четырёхходовой однобитный мультиплексор. Когда адресный вход sel=0, к выходу должен быть подключён вход d0, sel=1 – d1 и так далее.

```
module testquestion(
    input d0, d1, d2, d3,
    input [1:0] sel,
    output reg y
);
```

//пользовательский код

endmodule

№ 32

Используя приведённый шаблон модуля, реализовать с помощью оператора множественного выбора четырёхвыходной однобитный демультиплексор. Когда адресный вход sel=0, вход x должен быть подключён к выходу y0, когда sel=1 вход подключается к выходу y1 и так далее. На неиспользованных выходах должно быть установлено значение 0.

```
module testquestion(
    input [1:0] sel,
    input x,
    output reg y0, y1, y2, y3
);
```

//пользовательский код

endmodule

№ 33

Используя приведённый шаблон модуля, реализовать с помощью оператора множественного выбора трёхвыходной однобитный демультиплексор. Когда адресный вход sel=0, вход x должен быть подключён к выходу y0, когда sel=1 вход подключается к выходу y1 и так далее. На неиспользованных выходах должно быть установлено значение 0. Значение всех выходов по умолчанию должно быть равно 0.

```
module testquestion(
    input [1:0] sel,
    input x,
```

	<pre> output reg y0, y1, y2); //пользовательский код endmodule </pre>
№ 34	<p>Используя приведённый шаблон модуля, реализовать с помощью оператора выбора if/else двухвыходной однобитный демультиплексор. Когда адресный вход sel=0, вход x должен быть подключён к выходу y0, в противном случае вход подключается к выходу y1. На неиспользованных выходах должно быть установлено значение 0.</p> <pre> module testquestion(input sel, x, output reg y0, y1); //пользовательский код endmodule </pre>
№ 35	<p>Используя приведённый шаблон модуля, реализовать с помощью оператора множественного выбора дешифратор, формирующий из входного двухразрядного кода выходной унарный четырёхразрядный код.</p> <pre> module testquestion(input [1:0] x, output reg [3:0] y); //пользовательский код endmodule </pre>
№ 36	<p>Используя приведённый шаблон модуля, реализовать с помощью оператора битового сдвига дешифратор, формирующий из входного двухразрядного кода выходной унарный четырёхразрядный код.</p> <pre> module testquestion(input [1:0] x, output [3:0] y); //пользовательский код endmodule </pre>
№ 37	<p>Используя приведённый шаблон модуля, реализовать D-триггер. Синхронизация должна осуществляться по фронту тактового сигнала clk.</p> <pre> module testquestion(input d, clk, output reg y); //пользовательский код </pre>

№ 38	<p>endmodule</p> <p>1. Используя приведённый шаблон модуля, реализовать D-триггер. Синхронизация должна осуществляться по срезу тактового сигнала clk.</p> <pre> module testquestion(input d, clk, output reg y); //пользовательский код </pre>
№ 39	<p>endmodule</p> <p>Используя приведённый шаблон модуля, реализовать D-триггер с синхронным сбросом. Синхронизация должна осуществляться по фронту тактового сигнала clk. Если в момент фронта тактового сигнала nrst = 0, то в триггер необходимо записать значение 0.</p> <pre> module testquestion(input d, clk, nrst, output reg y); //пользовательский код </pre>
№ 40	<p>endmodule</p> <p>Используя приведённый шаблон модуля, реализовать D-триггер с синхронным сбросом. Синхронизация должна осуществляться по срезу тактового сигнала clk. Если в момент среза тактового сигнала nrst = 0, то в триггер необходимо записать значение 0.</p> <pre> module testquestion(input d, clk, nrst, output reg y); //пользовательский код </pre>
№ 41	<p>endmodule</p> <p>Используя приведённый шаблон модуля, реализовать D-триггер с асинхронным сбросом. Синхронизация должна осуществляться по фронту тактового сигнала clk. При срезе сигнала nrst в триггер необходимо записать значение 0.</p> <pre> module testquestion(input d, clk, nrst, output reg y); //пользовательский код </pre>
№ 42	<p>endmodule</p> <p>Используя приведённый шаблон модуля, реализовать D-триггер с асинхронным сбросом. Синхронизация должна осуществляться по фронту тактового сигнала clk. При срезе сигнала nrst в триггер необходимо записать значение 0.</p>

	<pre> module testquestion(input d, clk, nrst, output reg y); //пользовательский код endmodule </pre>
№ 43	<p>Используя приведённый шаблон модуля, реализовать счётный Т-триггер. Изменение значения триггера должно осуществляться по фронту тактового сигнала clk.</p> <pre> module testquestion(input clk, output reg y); //пользовательский код endmodule </pre>
№ 44	<p>1. Используя приведённый шаблон модуля, реализовать счётный Т-триггер. Изменение значения триггера должно осуществляться по срезу тактового сигнала clk.</p> <pre> module testquestion(input clk, output reg y); //пользовательский код endmodule </pre>
№ 45	<p>Используя приведённый шаблон модуля, реализовать счётный Т-триггер с асинхронным сбросом. Изменение значения триггера должно осуществляться по фронту тактового сигнала clk, а при срезе сигнала nrst в триггер необходимо записать значение 0.</p> <pre> module testquestion(input clk, nrst, output reg y); //пользовательский код endmodule </pre>
№ 46	<p>Используя приведённый шаблон модуля, реализовать счётный Т-триггер с асинхронным сбросом. Изменение значения триггера должно осуществляться по срезу тактового сигнала clk, а при срезе сигнала nrst в триггер необходимо записать значение 0.</p> <pre> module testquestion(input clk, nrst, </pre>

- output reg y
-);
- //ПОЛЬЗОВАТЕЛЬСКИЙ КОД
- endmodule
- Вопросы закрытого типа:*
- № 1 На каком из рисунков приведено RTL представление следующего модуля?
- ```

module testquestion(
 input clk, nrst,
 output reg [1:0] y
);
always @(posedge clk or negedge nrst) begin
 if (~nrst) y <= 2'b00;
 else y <= y + 2'b01;
end
endmodule

```
- <figure class="image"></figure>
- № 2 На каком из рисунков приведено RTL представление следующего модуля?
- ```


module testquestion(
    input clk, nrst,
    output reg [1:0] y
);
always @(posedge clk or negedge nrst) begin
    if (~nrst) y <= 2'b11;
    else y <= y + 2'b01;
end
endmodule

```
- <figure class="image"></figure>
- № 3 На каком из рисунков приведено RTL представление следующего модуля?
- ```


module testquestion(
 input clk, nrst,
 output reg [1:0] y
);
always @(posedge clk or negedge nrst) begin
 if (~nrst) y <= 2'b00;

```




- else y <= ~y;  
end  
endmodule
- № 4  На каком из рисунков приведено RTL представление следующего модуля?
- ```


module testquestion(
    input clk, nrst,
    output reg [1:0] y
);
always @(posedge clk) begin
    if (~nrst) y <= ~y;
    else y <= y + 2'b01;
end
endmodule

```
- № 5  На каком из рисунков приведено RTL представление следующего модуля?
- ```

module testquestion(
 input [3:0] a,
 input [3:0] b,
 output [3:0] y
);
assign y = a + b;
endmodule

```
- № 6  На каком из рисунков приведено RTL представление следующего модуля?
- ```

module testquestion(
    input [3:0] a,
    input [3:0] b,
    output [3:0] y
);
assign y = a << b[1:0];
endmodule

```
- 

№ 7 На каком из рисунков приведено RTL представление следующего модуля?

```
module testquestion(  
    input [3:0] a,  
    input [3:0] b,  
    output [3:0] y  
);  
  
assign y = a % b;  
  
endmodule
```

<figure class="image"></figure>

№ 8 На каком из рисунков приведено RTL представление следующего модуля?

```
module testquestion(  
    input [3:0] a,  
    input [3:0] b,  
    output [3:0] y  
);  
  
assign y = a * b;  
  
endmodule
```

<figure class="image"></figure>


№ 9 На каком из рисунков приведено RTL представление следующего модуля?

```
module testquestion(  
    input [3:0] x,  
    output y  
);  
  
assign y = &x;  
  
endmodule
```


<figure class="image"></figure>

№ 10 На каком из рисунков приведено RTL представление следующего модуля?







```
module testquestion(  
    input [3:0] x,  
    output y  
);  
  
assign y = ^x;  
  
endmodule
```

- № 11  На каком из рисунков приведено RTL представление следующего модуля?
- ```

module testquestion(
 input [3:0] x,
 output y
);
assign y = |x;
endmodule

```
- № 12  На каком из рисунков приведено RTL представление следующего модуля?
- ```

module testquestion(
    input [3:0] x,
    output y
);
assign y = ~|x;
endmodule

```
- № 13  С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- 
- A +
- Б -
- В *
- Г %
- № 14  С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- 
- A +
- Б -
- В *
- Г %
- № 15  С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- 

- А +
- Б -
- В *
- Г %
- № 16 С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- <figure class="image"></figure>
- А +
- Б -
- В *
- Г %
- № 17 С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- <figure class="image"></figure>
- А &&
- Б ||
- В &
- Г |
- № 18 С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- <figure class="image"></figure>
- А &&
- Б ||
- В &
- Г |
- № 19 С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- <figure class="image"></figure>
- А &&
- Б ||
- В &
- Г |
- № 20 С помощью какого оператора языка Verilog была синтезирована приведённая схема?
- <figure class="image"></figure>

- А &&
- Б ||
- В &
- Г |
- № 21 С помощью какого оператора языка Verilog была синтезирована приведённая схема, если первым операндом была шина а, а вторым - b.
- <figure class="image"></figure>
- А >
- Б >=
- В <
- Г <=
- № 22 С помощью какого оператора языка Verilog была синтезирована приведённая схема, если первым операндом была шина а, а вторым - b.
- <figure class="image"></figure>
- А >
- Б >=
- В <
- Г <=
- № 23 С помощью какого оператора языка Verilog была синтезирована приведённая схема, если первым операндом была шина а, а вторым - b.
- <figure class="image"></figure>
- А >
- Б >=
- В <
- Г <=
- № 24 С помощью какого оператора языка Verilog была синтезирована приведённая схема, если первым операндом была шина а, а вторым - b.
- <figure class="image"></figure>
- А >
- Б >=
- В <
- Г <=
- № 25 На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?
- module fsm(input clk, nrst, input [1:0] x, output reg y);
- parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10, Q3 = 2'b11;

```

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

    STATE <= Q0;

else

    case (STATE)

        Q0: if (x == 2'b01) STATE <= Q1;
        Q1: if (x == 2'b01) STATE <= Q2;
        Q2: if (x == 2'b01) STATE <= Q3;
        Q3: if (x == 2'b01) STATE <= Q0;

    endcase

always @(*)

case (STATE)

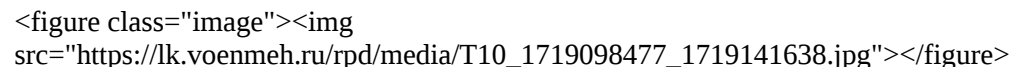
    Q0: y <= x[0];
    Q1: y <= x[1];
    Q2: y <= x[0];
    Q3: y <= x[1];

endcase

endmodule

```

№ 26


 На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```

module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10, Q3 = 2'b11;

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

    STATE <= Q0;

else

    case (STATE)

        Q0: if (x != 2'b01) STATE <= Q1;
        Q1: if (x != 2'b01) STATE <= Q2;
        Q2: if (x != 2'b01) STATE <= Q3;
        Q3: if (x != 2'b01) STATE <= Q0;

    endcase

endmodule

```

№ 27

```
endcase

always @(*)

case (STATE)

  Q0: y <= x[0];

  Q1: y <= x[1];

  Q2: y <= x[0];

  Q3: y <= x[1];

endcase

endmodule

<figure class="image"></figure>
На каком из рисунков приведен граф состояний, соответствующий конечному
автомату, описание которого на языке Verilog приведено ниже?

module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10, Q3 = 2'b11;

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

  STATE <= Q0;

else

  case (STATE)

    Q0: begin if (x == 2'b01) STATE <= Q1; if (x == 2'b10) STATE <= Q3; end

    Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b10) STATE <= Q0; end

    Q2: begin if (x == 2'b01) STATE <= Q3; if (x == 2'b10) STATE <= Q1; end

    Q3: begin if (x == 2'b01) STATE <= Q0; if (x == 2'b10) STATE <= Q2; end

  endcase

always @(*)

case (STATE)

  Q0: y <= x[0];

  Q1: y <= x[1];

  Q2: y <= x[0];

  Q3: y <= x[1];

endcase

endmodule

<figure class="image"></figure>
```

№ 28

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10, Q3 = 2'b11;

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

STATE <= Q0;

else

case (STATE)

Q0: if (x == 2'b01) STATE <= Q1;

Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b00) STATE <= Q0; end

Q2: begin if (x == 2'b01) STATE <= Q3; if (x == 2'b00) STATE <= Q0; end

Q3: if (x == 2'b01 | x == 2'b00) STATE <= Q0;

endcase

always @(*)

case (STATE)

Q0: y <= x[0];

Q1: y <= x[1];

Q2: y <= x[0];

Q3: y <= x[1];

endcase

endmodule
```

<figure class="image"></figure>

№ 29

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10;

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

STATE <= Q0;

else

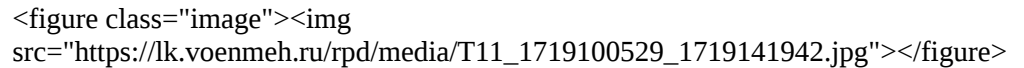
case (STATE)
```


№ 30

```
Q0: if (x == 2'b01) STATE <= Q1;
Q1: if (x == 2'b01) STATE <= Q2;
Q2: if (x == 2'b01) STATE <= Q0;
endcase

always @(*)
case (STATE)
Q0: y <= x[0];
Q1: y <= x[1];
Q2: y <= x[0];
endcase

endmodule
```



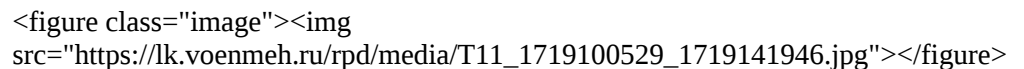
На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);
parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10;
reg [1:0] STATE;

always @(posedge clk or negedge nrst)
if (~nrst)
STATE <= Q0;
else
case (STATE)
Q0: begin if (x == 2'b01) STATE <= Q1; if (x == 2'b10) STATE <= Q2; end
Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b10) STATE <= Q0; end
Q2: begin if (x == 2'b01) STATE <= Q0; if (x == 2'b10) STATE <= Q1; end
endcase

always @(*)
case (STATE)
Q0: y <= x[0];
Q1: y <= x[1];
Q2: y <= x[0];
endcase

endmodule
```



№ 31

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10;

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

STATE <= Q0;

else

case (STATE)

Q0: begin if (x == 2'b01) STATE <= Q1; if (x == 2'b10) STATE <= Q2; end

Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b00) STATE <= Q0; end

Q2: if (x == 2'b00) STATE <= Q0;

endcase

always @(*)

case (STATE)

Q0: y <= x[0];

Q1: y <= x[1];

Q2: y <= x[0];

endcase

endmodule
```

<figure class="image"></figure>

№ 32

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 2'b00, Q1 = 2'b01, Q2 = 2'b10;

reg [1:0] STATE;

always @(posedge clk or negedge nrst)

if (~nrst)

STATE <= Q0;

else

case (STATE)

Q0: begin if (x == 2'b01) STATE <= Q1; if (x == 2'b10) STATE <= Q2; end

Q1: if (x == 2'b01) STATE <= Q2;
```

№ 33

```
Q2: if (x == 2'b00) STATE <= Q0;
endcase
always @(*)
case (STATE)
Q0: y <= x[0];
Q1: y <= x[1];
Q2: y <= x[0];
endcase
endmodule
```

<figure class="image"></figure>

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);

parameter Q0 = 3'b000, Q1 = 3'b001, Q2 = 3'b010, Q3 = 3'b011, Q4 = 3'b100;

reg [2:0] STATE;

always @(posedge clk or negedge nrst)
if (~nrst)
STATE <= Q0;
else
case (STATE)
Q0: if (x == 2'b01) STATE <= Q1;
Q1: if (x == 2'b01) STATE <= Q2;
Q2: begin if (x == 2'b01) STATE <= Q3; if (x == 2'b11) STATE <= Q4; end
Q3: if (x == 2'b01) STATE <= Q0;
Q4: STATE <= Q0;
endcase
always @(*)
case (STATE)
Q0: y <= x[0];
Q1: y <= x[1];
Q2: y <= x[0];
Q3: y <= x[1];
Q4: y <= x[0];
endcase
```

№ 34

endmodule

<figure class="image"></figure>

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);
```

```
parameter Q0 = 3'b000, Q1 = 3'b001, Q2 = 3'b010, Q3 = 3'b011, Q4 = 3'b100;
```

```
reg [2:0] STATE;
```

```
always @(posedge clk or negedge nrst)
```

```
if (~nrst)
```

```
STATE <= Q0;
```

```
else
```

```
case (STATE)
```

```
Q0: if (x == 2'b01) STATE <= Q1;
```

```
Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b11) STATE <= Q4; end
```

```
Q2: if (x == 2'b01) STATE <= Q3;
```

```
Q3: if (x == 2'b01) STATE <= Q0;
```

```
Q4: STATE <= Q0;
```

```
endcase
```

```
always @(*)
```

```
case (STATE)
```

```
Q0: y <= x[0];
```

```
Q1: y <= x[1];
```

```
Q2: y <= x[0];
```

```
Q3: y <= x[1];
```

```
Q4: y <= x[0];
```

```
endcase
```

```
endmodule
```

<figure class="image"></figure>

№ 35

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```
module fsm(input clk, nrst, input [1:0] x, output reg y);
```

```
parameter Q0 = 3'b000, Q1 = 3'b001, Q2 = 3'b010, Q3 = 3'b011, Q4 = 3'b100;
```

```
reg [2:0] STATE;
```

```
always @(posedge clk or negedge nrst)
```

```

if (~nrst)
    STATE <= Q0;
else
    case (STATE)
        Q0: if (x == 2'b01) STATE <= Q1;
        Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b11) STATE <= Q4; end
        Q2: begin if (x == 2'b01) STATE <= Q3; if (x == 2'b11) STATE <= Q4; end
        Q3: begin if (x == 2'b01) STATE <= Q0; if (x == 2'b11) STATE <= Q4; end
        Q4: STATE <= Q0;
    endcase
always @(*)
    case (STATE)
        Q0: y <= x[0];
        Q1: y <= x[1];
        Q2: y <= x[0];
        Q3: y <= x[1];
        Q4: y <= x[0];
    endcase
endmodule

```

№ 36

<figure class="image"></figure>

На каком из рисунков приведен граф состояний, соответствующий конечному автомату, описание которого на языке Verilog приведено ниже?

```

module fsm(input clk, nrst, input [1:0] x, output reg y);
    parameter Q0 = 3'b000, Q1 = 3'b001, Q2 = 3'b010, Q3 = 3'b011, Q4 = 3'b100;
    reg [2:0] STATE;
    always @(posedge clk or negedge nrst)
        if (~nrst)
            STATE <= Q0;
        else
            case (STATE)
                Q0: if (x == 2'b01) STATE <= Q1;
                Q1: begin if (x == 2'b01) STATE <= Q2; if (x == 2'b11) STATE <= Q4; end
                Q2: begin if (x == 2'b01) STATE <= Q3; if (x == 2'b11) STATE <= Q4; end
                Q3: begin if (x == 2'b01) STATE <= Q0; if (x == 2'b11) STATE <= Q4; end
            endcase
    endmodule

```

```

    Q4: if (x == 2'b11) STATE <= Q0;
endcase

always @(*)
case (STATE)
    Q0: y <= x[0];
    Q1: y <= x[1];
    Q2: y <= x[0];
    Q3: y <= x[1];
    Q4: y <= x[0];
endcase

endmodule

<figure class="image"></figure>

```