Белорусский государственный университет

УТВЕРЖДАЮ
Проректор по учебной работе
и образовательным инновациям

_____ О.И.Чуприс
(подпись)
20.06.2018
(дата утверждения)

Регистрационный № УД-6082/уч.

ВЕРИФИКАЦИЯ ЦИФРОВЫХ СИСТЕМ

Учебная программа учреждения высшего образования
по учебной дисциплине для специальности:

1-31 03 01 Математика (по направлениям)
Направление специальности
1-31 03 01-04 Математика (научно-конструкторская деятельность)

2018 г.

Учебная программа составлена на основе образовательного стандарта высшего образования ОСВО 1-31 03 01-2013 (30.08.2013) и учебного плана G31-209/уч. (29.05.2015) специальности 1-31 03 01 Математика (по направлениям), направление 1-31 03 01-04 Математика (научно-конструкторская деятельность).

Составитель:

Гладков А. Л. – заведующий кафедрой математической кибернетики механико-математического факультета Белорусского государственного университета, доктор физико-математических наук, профессор;

Зайцев В. С., старший преподаватель кафедры математической кибернетики механико-математического факультета Белорусского государственного университета.

Рекомендована к утверждению:

Кафедрой математической кибернетики механико-математического факультета Белорусского государственного университета

(протокол № 9 от 29.04.2018);

Учебно-методической комиссией механико-математического факультета Белорусского государственного университета

(протокол №8 от 19.06.2018).

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В процессе разработки цифровых устройств функциональная верификация играет очень важную роль, так как именно она отвечает на вопрос: «соответствует ли данное устройство спецификации?». Функциональная верификация проводится на стадии разработки описания будущего устройства и не требует прототипа, поэтому она позволяет проверить корректность работы и исправить ошибки в описании еще по начала производства реальных устройств. В настоящее время промышленным стандартом верификации является пакет UVM написанный на языке SystemVerilog. Использование этого пакета позволяет создавать конфигурируемые тестовые окружения, которые можно переиспользовать множество раз без глобальной переработки кодовой базы исходного тестового окружения. Такой подход экономит время и деньги компаний, позволяет ускорить процесс разработки интегральных микросхем.

Дисциплина «Верификация цифровых систем» знакомит студентов с особенностями языка верификации аппаратуры SystemVerilog, а также с пакетом UVM. Студентами будут изучены специализированные конструкции для описания случайных переменных, задания ограничений, сбора функционального покрытия и управления параллельными процессами. Также будут рассмотрены концепции наследования классов. Будут изучены стандартные классы пакета UVM как uvm_driver, uvm_monitor, uvm_agent, uvm_env, uvm_test, uvm_sequence_item, uvm_sequnce, которые используются для организации тестовых окружений для сложных систем. Также будет изучен паттерн проектирования с использование портов передачи транзакций в системе, на основе классов пакета uvm_tlm_port.

В рамках практических занятий по данной дисциплине, студенты вспомнят базовые конструкции языка, и изучат расширенные возможности и паттерны проектирования языка верификации аппаратуры - SystemVerilog. Более того, студентами будут разработаны тестовые окружения для проверки некоторых простых интерфейсов передачи данных, таких как UART, SPI и APB. Разработан набор тестовых сценариев для этих интерфейсов, изучен метод использования регистровой модели.

Для понимания учебного материала требуется знание основных приемов и подходов к написанию программ из дисциплины «методы программирования и информатика» и некоторые факты из дисциплины «теория булевых функций». Студент также должен обладать минимальным опытом программирования вне зависимости от языка программирования. Материал данной дисциплины также тесно связан с дисциплиной «Языки описания программно-аппаратных систем», которая читается параллельно, а также является прямым продолжением материала программы «Языки верификации аппаратуры», который читается на 3-м курсе.

Цель дисциплины: получение расширенных знаний о языке верификации аппаратуры SystemVerilog, специальных конструкциях. Также обучающиеся должны освоить использование классов из пакета UVM

(universal verification methodology - Универсальная методология верификации) и освоить построение тестовых окружений и тестовых сценариев на их основе.

Развивающая цель: получения студентами практических навыков работы с языком верификации аппаратуры SystemVerilog и пакетом UVM. Усовершенствование навыков студентов в области построения тестовых окружений для отладки цифровых схем и поиска ошибок в описаниях цифровых схем.

Воспитательная цель: развитие у студентов навыка самостоятельного поиска и получения информации в литературе и сети интернет, который в будущем может упростить им решение возникающих задач в проектировании и в дальнейшей их повседневной жизни.

Задача дисциплины: изучение расширенного синтаксиса, типов данных и специализированных конструкций языка SystemVerilog. Изучение концепций классов пакета верификации UVM.

В результате изучения дисциплины "Верификация цифровых систем" студент должен:

знать: Синтаксис языка SystemVerilog. Встроенные функции и операторы языка SystemVerilog. Встроенные конструкции ограничений для случайных переменных. Встроенные конструкции для сбора функционального покрытия. Встроенные примитивы синхронизации процессов. Структуру и функции расширяемого драйвера, монитора и агента. Структуру тестов окружения и тестового сценария. Подходы к верификации цифрового блока с использованием регистровой модели.

уметь: компилировать исходные файлы проекта. Запускать симуляцию проекта. Использовать классы UVM пакета, проектировать тестовое окружение, тестовые сценарии, использовать регистровую модель, использовать метрики покрытия цифрового блока, оценивать полноту и качество проведенной верификации.

Учебная дисциплина строится таким образом, чтобы обучающийся приобрел следующие компетенции специалиста:

Академические:

АК-1. Уметь применять базовые научно-теоретические знания для решения теоретических и практических задач.

АК-2. Владеть системным и сравнительным анализом.

АК-3. Владеть исследовательскими навыками.

АК-4. Уметь работать самостоятельно.

АК-5. Быть способным вырабатывать новые идеи (обладать креативностью).

АК-6. Владеть междисциплинарным подходом при решении проблем.

АК-7. Иметь навыки, связанные с использованием технических устройств, управлением информацией и работой с компьютером.

АК-8. Обладать навыками устной и письменной коммуникации.

АК-9. Уметь учиться, повышать свою квалификацию в течение всей жизни.

Социально-личностные:

СЛК-2. Быть способным к социальному взаимодействию.

СЛК-3. Обладать способностью к межличностным коммуникациям.

СЛК-5. Быть способным к критике и самокритике.

СЛК-6. Уметь работать в команде.

Профессиональные:

ПК-1. Разрабатывать практические рекомендации по использованию научных исследований, планировать и проводить экспериментальные исследования, исследовать патентоспособность и показатели технического уровня разработок программного обеспечения информационных систем.

ПК-4. Разрабатывать и тестировать информационные системы, осуществлять защиту приложений и данных.

ПК-9. Осуществлять выбор оптимального варианта проведения научно-исследовательских работ.

ПК-13. Взаимодействовать со специалистами смежных профилей.

ПК-16. Готовить доклады, материалы к презентациям.

ПК-22. Осваивать и реализовывать управленческие инновации в сфере высоких технологий.

ПК-27. Реализовывать инновационные проекты в профессиональной деятельности.

Дисциплина «Верификация цифровых систем» относится к дисциплине специализации.

Учебная программа по дисциплине «Верификация цифровых систем» предназначена для студентов 4 курса (7 семестр) очной формы получения высшего образования.

На изучение учебной дисциплины по специальности 1-31 03 01 «Математика (по направлениям)» (1-31 03 01-01 Математика (научно-конструкторская деятельность» предусмотрено всего 124 часа, в том числе 66 часов аудиторных занятий, включая 34 часа лекций, 26 часов лабораторных занятий, 6 часов УСР. Текущая аттестация – экзамен.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1. Знакомство с понятием верификации цифровых схем.

Место верификации в процессе разработки цифровых устройств. Задачи верификации. Основные подходы при проведении верификации. Языки верификации аппаратуры. Симуляторы, запуск моделирования.

Тема 2. Основы языка SystemVerilog.

Основные операторы, типы данных, задачи и функции, конструкции создания циклов и условий, системные функции, классы, наследование классов, модули, программы. Интерфейсы, виртуальные интерфейсы. Запуск параллельных задач. Управление сигналами интерфейса в объекте класса.

Тема 3. Случайные переменные и функциональное покрытие.

Случайные переменные с равномерным и нормальным распределением. Задание ограничений для случайных переменных. Условия и циклы в ограничениях. Стандартные функции ограничений. Группы покрытия, их стандартные функции и опции. Задание точек покрытия. Описание корзин покрытия для множества значений, диапазонов и последовательностей. Условия покрытия корзин.

Тема 4. Базовые понятия пакета UVM.

Моделирование на уровне транзакций. Иерархия UVM классов. Макросы регистрации классов и полей. Фазы выполнения теста. Фазы создания теста. Добавление пользовательских фаз. Управление переходом между фазами. Средства UVM для вывода сообщений.

Тема 5. Создание UVM окружения.

Реализация транзакции, сиквенса, объекта конфигурации, драйвера, монитора, сиквенсера, агента, окружения, теста, библиотеки тестов, верхний уровень запуска тестового окружения. UVM TLM библиотека передачи транзакций. Порты передачи транзакций. Иерархия классов. Реализация блока анализа результатов.

Тема 6. Написание и управление тестовыми воздействиями

Изучение классов пакета UVM верхнего уровня, включающий в себя все необходимые компоненты для проверки цифрового блока: методы транзакций, сиквенсы, макросы для генерации и запуска транзакции, запуск сиквенсы по умолчанию, генерация последовательностей транзакций, интерфейс связи сиквен-драйвер-сиквенсер, сиквенсы с обратной связью, запуск нескольких сиквенсов в тесте, управление процессом выполнения сиквенсов, виртуальные сиквенсы, тесты, параметры командной строки, управление временем выполнения.

Тема 7. Использование UVM регистровой модели.

Обзор UVM регистровой модели. Автоматическое, явное, пассивное предсказание. Шаги интеграции регистровой модели. Адаптер регистровой модели. Подключение регистровой модели. Встроенные сиквенсы и тесты.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов					Количество часов УСР	Форма контроля знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
1	**Знакомство с понятием верификации цифровых схем.**	1						*Устный опрос*
2	**Основы языка SystemVerilog**	3			4			*Тест*
3	**Случайные переменные и функциональное покрытие.**	6			4			*Тест*
4	**Базовые понятия UVM.**	6			4		2	*Защита лабораторных работ*
5	**Создание UVM окружения.**	6			6		2	*Защита лабораторных работ*
6	**Написание и управление тестовыми воздействиями**	6			4		2	*Защита лабораторных работ*
7	**Использование UVM регистровой модели.**	6			4			*Тест*
	ИТОГО	34			26		6	

ИНФОРМАЦИОННО - МЕТОДИЧЕСКАЯ ЧАСТЬ

Рекомендуемая литература

Основная:

1. 1800-2009 IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language.
2. C. Spear, G. Tumbush, SystemVerilog for Verification: A Guide to Learning the Testbench Language Features 3rd edition - San Francisco: Springer, 2012
3. SimHard | Верификация Цифровых Схем [Электронный ресурс] . - 2018 - Режим доступа: http://simhard.com/wiki/ - Дата доступа: 01.03.2018.
4. UVM tutorial for candy lovers | cluelogic.com [Электронный ресурс] . - 2018 - Режим доступа: http://cluelogic.com/2013/02/uvm-tutorial-for-candy-lovers-register-access-methods/ - Дата доступа: 01.03.2018.
5. UVM Cookbook. | Chapter "Register Abstraction Layer" [Электронный ресурс] . - 2018 - Режим доступа: https://verificationacademy.com/cookbook/registers - Дата доступа: 01.03.2018.
6. SystemVeriloig Golden Reference Guide - A concise guide to SystemVeriloig IEEE Std 1800-2009 Version 5.0, August 2012, Doulos Ltd
7. Хаханов В.И. Проектирование и верификация цифровых систем на кристаллах. Verilog & System Verilog / В.И. Хаханов, И.В. Хаханова, Е.И. Литвинова, О.А. Гузь. – Харьков : ХНУРЭ. – 2010. – 528 с.

Дополнительная:

1. ASIC-world | SystemVerilog [Электронный ресурс]. - 2018 - Режим доступа: http://www.asic-world.com/systemverilog - Дата доступа: 01.03.2018.
2. Verification Academy | UVM [Электронный ресурс]. - 2018 - Режим доступа: https://verificationacademy.com/verification-methodology-reference/uvm/docs_1.2/html/index.html - Дата доступа: 01.03.2018.
3. Testbench | SystemVerilog [Электронный ресурс]. - 2018 - Режим доступа: http://www.testbench.in/ - Дата доступа: 01.03.2018.
4. Verification Academy | UVM [Электронный ресурс]. - 2018 - Режим доступа: https://verificationacademy.com/cookbook/sequences/sequencelibrary - Дата доступа: 01.03.2018.

5. Verification Academy | UVM [Электронный ресурс]. - 2018 - Режим доступа: https://verificationacademy.com/verification-methodology-reference/uvm/docs_1.2/html/ - Дата доступа: 01.03.2018.
6. Verification Academy | UVM [Электронный ресурс]. - 2018 - Режим доступа: https://verificationacademy.com/cookbook/driver/sequence_api - Дата доступа: 01.03.2018.

7. Testbench.in | Functional coverage [Электронный ресурс]. - 2018 - Режим доступа: http://www.testbench.in/CO_00_INDEX.html - Дата доступа: 01.03.2018.
8. Testbench.in | Randomization [Электронный ресурс]. - 2018 - Режим доступа: http://www.testbench.in/CR_00_INDEX.html - Дата доступа: 01.03.2018.
9. A practical look | SystemVerilog Coverage [Электронный ресурс]. - 2018 - Режим доступа: http://www.slideshare.net/obsidiansoft/doulos-coveragetipstricks - Дата доступа: 01.03.2018.
10. SmartDv verification | Functional Coverage [Электронный ресурс]. - 2018 - Режим доступа: http://www.asic-world.com/systemverilog/coverage.html - Дата доступа: 01.03.2018.

Организация управляемой самостоятельной работы студентов

Основным критерием оценки результатов аудиторной самостоятельной работы обучающихся по дисциплине является уровень усвоения учебного материала, который проверяется и оценивается в процессе выполнения лабораторных работ.

Управляемая самостоятельная работа (УСР) студентов – это самостоятельная работа, выполняемая по заданию преподавателя, при его методическом руководстве и контроле.

Основными направлениями управляемой самостоятельной работы в овладении знаниями учебной дисциплины «Верификация цифровых систем» являются:

- изучение материалов лекций и слайдов, размещенных на образовательном ресурсе университета;
- ознакомление с программой учебной дисциплины;
- ознакомление со списком рекомендуемой литературы по дисциплине в целом, изучение необходимой литературы по теме, подбор дополнительной литературы в сети интернет;
- получение навыков разработки в ходе выполнения самостоятельных практических заданий, а также выполнения тестов контроля знаний, размещенных в сети интернет;
- подготовка к экзамену.

Материалы и задания, размещенные в сети интернет, подразумевают в дальнейшем переход на полное самостоятельное изучение материала, согласно главам в плане. В этом случае студент сможет получить знания главным образом за счет самостоятельной работы.

Учет результатов контроля текущей успеваемости студентов ведется преподавателем. Полученные студентом количественные результаты УСР учитываются как составная часть итоговой оценки по дисциплине в рамках рейтинговой системы.

Перечень используемых средств диагностики результатов учебной деятельности

С целью текущего контроля знаний студентов предусматривается проведение устного опроса, защиты лабораторных работ и выполнения тестовых заданий.

Методика формирования итоговой оценки

Итоговая оценка формируется на основе 3-х документов:

1.Правила проведения аттестации (Постановление № 53 от 29.05.2012 г.).

2.Положение о рейтинговой системе БГУ (ред.2015 г.).

3.Критерии оценки студентов (10 баллов).

Рекомендуемый перечень заданий лабораторных занятий и УСР

Тема 1.

1. Вывести в консоль значения литералов в десятичном, двоичном, шестнадцатиричном и строковом форматах.
2. Создать тип данных структура, содержащую 4 различных поля. Поля должны иметь тип целочисленный, строковый, бит-вектор. Создать две переменные полученного типа. Проинвертировать все битовые поля в первой переменной. Значение всех полей первой и второй переменных типа структуры вывести в консоль.
3. Объявить и заполнить строковую переменную. Вывести на экран первый и последний элемент строки. Длина строки должна быть больше 10 символов.
4. Объявить и заполнить строковую переменную. Длина строки должна быть больше 30 символов. Строка содержит символы пробела, разделяющие строку на слова, слов более одного. Создать массив строковых переменных и заполнить их словами из исходной переменной.
5. Создать 2-х мерный массив целочисленных переменных, заполнить случайными значениями и после отсортировать в порядке возрастания элементы в каждой строке.
6. Создать упакованный массив данных, в который поместить значения из предыдущего задания и вывести в консоль все элементы друг за другом без пробелов.
7. Создать динамический массив байт размером 10 кБ. Создать еще один динамический массив целых чисел подходящего размера и поместить в него значения первого. Второй массив не должен занимать более 10 кБ.
8. Создать ассоциативный массив из 1024 элементов, каждый элемент которого представляет собой динамический массив размером 1024 бит + номер. Номер определяется так, добавляем в ассоциативный массив первый элемент номер равен 1, второй 2, и т.д. Значения векторов в динамическом массиве заполнить случайным образом.
9. Создать очередь, каждый элемент которой представляет собой структуру данных из 2-го задания. Заполнить очередь 10 значениями. После удалить первые 3 элемента из очереди и вывести значения полей элементов в консоль.
10. Объявить класс, содержащий 3 переменные A, minAi, maxAi. A-массив целых чисел, minAi - минимальное значение, maxAi - максимальное значение в массиве, создать методы класса возвращающие максимальное и минимальное значение в массиве. А также методы вывода значений всех переменных в консоль, функция должна быть виртуальной. Массив заполнить 10 случайными значениями. Создать объект класса и вывести значения переменных в консоль.
11. Объявить класс, наследуемый от класса в предыдущем задании и переопределить функцию вывода значений всех переменных. Добавить к

существующему выводу, вывод сообщения об разработчике класса (ФИО латиницей).

12. Объявить пакет, который будет содержать объявления классов из двух предыдущих заданий. Объявить еще один класс в модуле, наследуемый от одного из классов описанных в пакете. Использовать импорт пакета. В наследуемом классе переопределить еще раз метод вывода сообщений, убрав вывод информации о разработчике.
13. Объявить класс, содержащий задачу, выводящую значения 1, 2, 3, ... и т.д. с периодичностью 1 us симуляционного времени. Объявить еще одну задачу, которая выводит текстовые сообщения "1000us", "2000us", "3000us" ... с периодичностью 1 ms. Объявить третью задачу, которая выводит сообщение "Start check processing..." после того как в первой задаче значения превысили 10000, и выводит сообщение "End check processing..." после того как вторая задача выведет сообщение "5000us". Запустить три задачи в параллель. Провести моделирование 100 ms.
14. Описать генератор тактового сигнала используя оператор цикла *while*.
15. Объявить и заполнить два ассоциативных массива произвольными данными. Реализовать функцию сравнения этих массивов.
16. Создать очередь целых чисел. Реализовать функции поиска максимального и минимального элементов.
17. Создать очередь массивов фиксированной длины, каждый из которых содержит два элемента. Элементы будут задавать нижнюю и верхнюю границы диапазона значений. Реализовать функцию проверки того, что заданные диапазоны во всех элементах очереди не пересекаются между собой.
18. Реализовать базовый класс A с полем data (динамический массив) и наследуемый от него класс B. Реализовать класс C, содержащий произвольную функцию (например, для подсчета суммы элементов массива data) с входным параметром типа класс A и далее вызвать эту функцию, передав в качестве аргумента объект типа класс B. Для этого следует правильно передать и обработать поле data объекта типа класс B в реализуемой функции класса C.
19. Реализовать интерфейс, параметризируемый двумя значениями: ширина адреса и ширина данных.

Тема 2.

1. Записать ограничения на рандомизацию параметров транзакции в соответствии с требованиями спецификации. Спецификация :
    - Транзакция содержит поля data, addr, strob, mode
    - Ширина данных 32 бита (data)
    - Ширина адреса 32 бита (addr)
    - Для поля Addr используется побайтовая адресация
    - Ширина строба 4 бита (strob), каждый строб отвечает за один байт в data
    - Поле mode имеет четыре значения Чтение, Запись, Ожидание, Ошибка (кодирование выбрать самостоятельно)

- Поле strob имеет только непрерывное заполнение единицами, т.е. значения 0000, 1010, 1101 и др. недопустимы
- Поле data принимает значения 0101….01 и 1010….10 с вероятностью 10% каждое, выпадение остальных значений равновероятно
- Поле addr меньше переменной max_addr и больше min_addr
- В режиме чтения адрес должен быть выровнен на 2
- Если значение поля mode равно Запись, то addr выравнен на 4

2. Записать ограничения на рандомизацию параметров транзакции в соответствии с требованиями спецификации. Спецификация :
   - Транзакция содержит поля data, addr, strob, mode, status
   - Ширина данных 128 бита (data)
   - Ширина адреса 32 бита (addr)
   - Ширина строба (strob) равна числу байт в шине данных
   - Поле mode имеет четыре значения Чтение, Запись, Ожидание, Ошибка
   - Поле strob имеет только непрерывное заполнение единицами, т.е. значения 0000, 1010, 1101 и др. недопустимы
   - Поле data принимает значения 0101….01 и 1010….10 с вероятностью 10%
   - Поле data принимает максимальное и минимальное значения с вероятностью 1%
   - Поле addr меньше переменной max_user_addr и меньше максимально возможного MAX_BUS_ADDR

   addr должен иметь значения адресующие только по 8 байт, если значение поля mode равно Чтение
   - Заполнение 1 поля strob должно начинаться с бита, номер которого равен значению младших 3-разрядов адреса.
   - Поле статус всегда равно OK, если mode равен Запись, иначе UNDEF
3. Записать ограничения на рандомизацию параметров транзакции в соответствии с требованиями спецификации. Спецификация :
   - Транзакция содержит поля data, addr, strob, mode, status, delay_en, write_delay, read_delay
   - Ширина данных 64 бита (data)
   - Ширина адреса 32 бита(addr)
   - Ширина строба (strob) равна числу байт в шине данных
   - Поле mode имеет четыре значения Чтение, Запись, Ожидание, Ошибка
   - Поле strob имеет только непрерывное заполнение единицами
   - Поле data принимает значения 0101….01 и 1010….10 с вероятностью 5%
   - Поле data принимает максимальное и минимальное значения с вероятностью 1%

- Поля с именем *delay принимают целые значения большие или равные 0
- Если значение поля delay_en равно 0, то переменные write_delay, read_dealy равны 0.
- Значение поля write_delay меньше 10, если dealy_en равен 1
- Значение поля read_delay меньше 200, если dealy_en равен 1

4. Записать ограничения на рандомизацию параметров транзакции в соответствии с требованиями спецификации. Спецификация :
   - Транзакция содержит поля data, addr, strob, mode
   - Ширина данных 64 бита, поле data - очередь из N данных
   - Ширина адреса 32 бита, поле addr - очередь из N адресов
   - Ширина строба равна числу байт в шине данных, strob - очередь из N стробов
   - Число элементов в очередях адреса, данных и стробов должно совпадать и должно быть не более 16.
   - Поле mode имеет два значения Чтение, Запись
   - Поле strob имеет только непрерывное заполнение единицами, с учетом очереди слов (т.е. Множество единиц в словах составленных друг за другом не должны перемешиваться 0, другими словами все стробы со 2-го и до предпоследнего всегда равны всем 1, если 1-й и последний стробы больше 0)
   - Поле data принимает значения 0101….01 и 1010….10 с вероятностью 25%
   - Поле data принимает максимальное и минимальное значения с вероятностью 6%
5. Задать группу покрытия для сбора функционального покрытия в соответствии с требованиями плана верификации (В конце моделирования вывести процент покрытия ковер-группы). Группа покрытия должна покрывать множество тестов представленных массивом строк.
6. Группа покрытия должна содержать точку покрытия, в которой покрываются последовательности событий: Проверить что было чтение по адресу 0,1,2,3,4,F после записи по этим же адресам, по-очереди по каждому адресу. Проверить, что во время команды чтения был произведен сброс, затем команда чтения, затем снова сброс.
7. Сделать ковер-группу, которая проверяет, что все размеры очереди в режиме чтения и записи транзакции из Спецификации 4 были проверены в тесте.

Тема 3.

1. По спецификации интерфейса (*) блока разработать транзакцию, позволяющую описать все доступные операции на заданном интерфейсе:с использованием UVM макросов/без использования UVM макросов.

2. Создать пакет, реализующий весь функционал агента (сиквенсер, драйвер, монитор) из существующих файлов описания всех составных блоков и скомпилировать его. (Можно использовать файлы из проекта https://www.edaplayground.com/x/3ru7)
3. Реализовать объект конфигурации агента, который содержит методы настройки в три режима: по умолчанию активный ведущий, пассивный, активный ведомый режимы
4. Реализовать фазу сброса в драйвере
5. Реализовать фазу, предшествующую фазе сброса
6. Подменить драйвер в агенте, из окружения в фазе создания компонент
7. Создать домен фазы выполнения для агента и подключить агент к этому домену

Тема 4.

1. Создать пакет, реализующий весь функционал агента из существующих файлов описания всех составных блоков и скомпилировать его.
2. По готовому описанию интерфейса и транзакции, разработать драйвер, позволяющую выполнять базовые операции на заданном интерфейсе.
3. По готовому описанию интерфейса и транзакции, разработать монитор, позволяющую отправить базовые операции на заданном интерфейсе.
4. Собрать агент из готовых компонентов (driver, monitor, sequencer, coverage_collector), реализовать фазы создания и подключения в зависимости от параметров объекта конфигурации.
5. Разработать блок сравнения пары транзакций.
6. Разработать блок сравнения произвольной последовательности данных
7. Реализовать окружение, в котором будет два APB агента, отличающихся шириной шины адреса и данных (адрес/данные 11/32 , 20/64).
8. Реализовать окружение, в котором собраны два APB агента, один в режиме мастер, другой в режиме ведомый, и запустить 100 случайных транзакций для передачи из одного агента во второй.

Тема 5.

1. Подготовить транзакцию, содержащую поля адреса и данных. Для адреса задан запрещенный диапазон значений, для данных - список запрещенных значений.
2. Подготовить транзакцию, в которой сравнение заданных полей отключено.
3. Подготовить сиквенс, генерирующую случайный адрес чтения из заданного диапазона с заданным шагом.
4. Подготовить сиквенс, генерирующую случайный адрес чтения ОЗУ, которая будет покрывать множество заданных диапазонов, попадая в каждый диапазон с заданной частотой.
5. Подготовить виртуальную сиквенс и сиквенсер генерирующую два потока данных с заданным приоритетом выполнения для каждого сиквенсера 10% и 90% соответственно.
6. Подготовить виртуальную сиквенс, которая случайно запускает одну из 10 сиквенсов на случайно выбранных 2 из 5 сиквенсерах.

7. Параметризовать тест аргументами командной строки для запуска различных сиквенсов.

Тема 6.

1. Описать регистровую модель по заданной спецификации тестируемого модуля.
2. Разработать тест по заданному сценарию с использованием регистровой модели.
3. Создать и подключить пользовательский предиктор регистровой модели. Добавить опцию очистки одного регистра при изменении бита в другом используя встроенные функции.
4. Разработать адаптер регистровой модели агента реализующего SPI интерфейс
5. Разработать адаптер регистровой модели агента реализующего AHB интерфейс

Рекомендуемый перечень вопросов к экзамену по дисциплине «Верификация цифровых систем»

1. Какие операторы в SystemVerilog используются для организации циклов?
2. Перечислить типы UVM компонентов, используемых в uvm_agent.
3. Описать класс, в котором создать очередь целых чисел случайной длины. Реализовать функцию заполнения первых 5 элементов значениями от 1 до 20, остальных - случайным образом.
4. Какие виды ограничений (constraint) доступны в SystemVerilog?
5. Для чего применяется оператор $cast?
6. Задать группу для сбора функционального покрытия последовательности значений поля data транзакции из задания 3. Должны проверяться покрытие последовательностей из 2, 4, 6, 8 единиц для поля data.
7. Какие типы данных доступны в SystemVerilog?
8. Какие встроенные методы можно использовать для ассоциативных массивов в языке SystemVerilog?
9. Классы в SystemVerilog?
10. Какие встроенные методы можно использовать для работы с динамическими массивами в языке SystemVerilog?
11. Описать операторы запуска параллельного выполнения в SystemVerilog.
12. Какие встроенные методы можно использовать для очереди в языке SystemVerilog?
13. Что такое интерфейс в SystemVerilog?
14. Возможно ли в SystemVerilog отключать режим рандомизации для отдельных переменных? Если да, то как?
15. Описать класс транзакции, содержащий два поля - dir, data и сгенерировать 10 транзакций. Каждую транзакцию генерировать после события "передний фронт" на тактовом генераторе из задания 3.

16. Что такое группы покрытия в SystemVerilog? Для чего они используются и как описываются?
17. Раскрыть понятие транзакции в UVM методологии верификации.
18. В каких случаях допускается использование функций в ограничениях?
19. Какие виды массивов реализованы в SystemVerilog?
20. Какие операторы позволяют запустить процессы в параллель? Опишите их особенности.
21. Чем отличается ассоциативные массивы от динамических?
22. Какие UVM компоненты, используемые в uvm_agent, не содержат предопределенного порта?
23. Допускается ли использование функций и циклов в ограничениях?
24. Какие операторы используются для назначения события?
25. Как задать список чувствительности и события для проверки групп покрытия?
26. Какие операторы используются для мониторинга событий?
27. Как реализуется проверка последовательности событий в группе покрытия?
28. Допускается ли наследование классов в языке SystemVerilog? Если да - какие типы наследования допускаются, если нет - почему.
29. Описать класс транзакции, содержащий поля data, addr. Сгенерировать две транзакции и сравнить их.
30. В чем отличие при использовании оператора if и оператора импликации при задании ограничений?
31. Чем отличается push-сиквенсер (класс uvm_push_sequenser) от обычного сиквенсера (класса uvm_sequencer)?
32. Какие типы данных могут быть использованы для создания упакованных массивов?
33. Описать механизм обмена транзакциями между драйвером и сиквенсером. Какие порты для этого используются?
34. Какой из встроенных методов нужно вызвать, чтобы преобразовать строку string str = "def0" в число типа integer?
35. Что представляет собой класс uvm_domain и для чего он применяется?

ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ

Название учебной дисциплины, с которой требуется согласование	Название кафедры	Предложения об изменениях в содержании учебной программы УВО по изучаемой учебной дисциплине	Решение, принятое кафедрой, разработавшей учебную программу (с указанием даты и номера протокола)
Цифро-аналоговые устройства	Кафедра математической кибернетики	Нет	Оставить содержание учебной программу без изменения. (протокол № 9 от 29.04.2018)

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ
К УЧЕБНОЙ ПРОГРАММЕ УВО ПО ИЗУЧАЕМОЙ УЧЕБНОЙ ДИСЦИПЛИНЕ на _____/_____ учебный год

№ п/п	Дополнения и изменения	Основание

Учебная программа пересмотрена и одобрена на заседании кафедры «Математической кибернетики» (протокол № от 201__ г.)

Заведующий кафедрой
д-р ф.-м. н., профессор _____ А.Л. Гладков
(подпись)

УТВЕРЖДАЮ
Декан факультета
к. ф.-м. н., доцент _____ Д.Г. Медведев
(подпись)