МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

УТВЕРЖДАЮ
Директор ИЭиАС
_____ С.И. Лукьянов

26.02.2020 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

ОСНОВЫ ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННОЙ КОМПОНЕНТНОЙ БАЗЫ

Направление подготовки (специальность)
11.03.04 Электроника и наноэлектроника

Направленность (профиль/специализация) программы
Программирование и электроника информационных систем

Уровень высшего образования - бакалавриат

Форма обучения
заочная

Институт/ факультет	Институт энергетики и автоматизированных систем
Кафедра	Электроники и микроэлектроники
Курс	5
Семестр	

Магнитогорск
2020 год

Рабочая программа составлена на основе ФГОС ВО по направлению подготовки 11.03.04 Электроника и наноэлектроника (уровень бакалавриата) (приказ Минобрнауки России от 19.09.2017 г. № 927)

Рабочая программа рассмотрена и одобрена на заседании кафедры Электроники и микроэлектроники
13.02.2020 г. протокол № 6

Зав. кафедрой _____ С.И. Лукьянов

Рабочая программа одобрена методической комиссией ИЭиАС
26.02.2020 г. протокол № 5

Председатель _____ С.И. Лукьянов

Рабочая программа составлена:
доцент кафедры ЭиМЭ, канд. техн. наук _____ Н.В. Швидченко

Рецензент:
директор СЦ ООО "ТЕХНОАП Инжиниринг", канд. техн. наук
_____ Е.С. Суспицын

Лист регистрации изменений и дополнений

№ п/п	Раздел программы	Краткое содержание изменения/дополнения	Дата. № протокола заседания кафедры	Подпись зав. кафедрой
1	Раздел 8	Актуализация перечня основной, дополнительной литературы и лицензионного программного обеспечения	31.08.2020 г., протокол № 1	

Лист актуализации рабочей программы

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2020 - 2021 учебном году на заседании кафедры Электроники и микроэлектроники

Протокол от 31.08.2020 г. № 1
Зав. кафедрой _____ С.И. Лукьянов

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2021 - 2022 учебном году на заседании кафедры Электроники и микроэлектроники

Протокол от __ _____ 20__ г. № __
Зав. кафедрой _____ С.И. Лукьянов

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2022 - 2023 учебном году на заседании кафедры Электроники и микроэлектроники

Протокол от __ _____ 20__ г. № __
Зав. кафедрой _____ С.И. Лукьянов

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2023 - 2024 учебном году на заседании кафедры Электроники и микроэлектроники

Протокол от __ _____ 20__ г. № __
Зав. кафедрой _____ С.И. Лукьянов

1 Цели освоения дисциплины (модуля)

Целями освоения дисциплины «Основы проектирования электронной компонентной базы» являются изучение современных методов и маршрутов проектирования электронной компонентной базы, средств и способов автоматизации процесса проектирования, проведение проектных расчетов и технико-экономическое обоснование принимаемых решений.

2 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина Основы проектирования электронной компонентной базы входит в часть учебного плана формируемую участниками образовательных отношений образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения дисциплин/ практик:

Дискретная математика

Информатика и информационные технологии

Физика конденсированного состояния

Физика

Математика

Элементы цифровой техники

Схемотехника

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для изучения дисциплин/практик:

САПР устройств промышленной электроники

Основы технологии электронной компонентной базы

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Основы проектирования электронной компонентной базы» обучающийся должен обладать следующими компетенциями:

Код индикатора	Индикатор достижения компетенции
ПК-1 Способен разрабатывать структурные и функциональные схемы электронных систем и комплексов, принципиальных схем устройств с использованием средств компьютерного проектирования, проведением проектных расчетов и технико-экономическим обоснованием принимаемых решений	
ПК-1.1	Разрабатывает эскизный проект, включающий: выбор структурной схемы электронного устройства или системы путем сопоставления различных вариантов и их оценки с точки зрения технических и экономических требований; рассчитывает все необходимые показатели структурной схемы электронного устройства или системы, в том числе показатели качества; выбирает и обосновывает схемы вспомогательных устройств
ПК-1.2	Производит технико-экономическое обоснование принятого решения с расчетами себестоимости устройства и стоимости его эксплуатации; сравнивает с аналогами по технико-экономическим характеристикам

4. Структура, объём и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 3 зачетных единиц 108 акад. часов, в том числе:

– контактная работа – 10,7 акад. часов:

– аудиторная – 10 акад. часов;

– внеаудиторная – 0,7 акад. часов

– самостоятельная работа – 93,4 акад. часов;

– подготовка к зачёту – 3,9 акад. часа

Форма аттестации - зачет

Раздел/ тема дисциплины	Курс	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа студента	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код компетенции
		Лек.	лаб. зан.	практ. зан.				
1. Основы проектирования								
1.1 Современная электронная компонентная база. Классификация. Область	5				10	Самостоятельное изучение учебной литературы	Контрольная работа	ПК-1.1, ПК-1.2
1.2 Проектирование электронной компонентной базы: основные этапы и уровни проектирования		1			10	Самостоятельное изучение учебной литературы	контрольная работа	ПК-1.1, ПК-1.2
1.3 Системы автоматизированного проектирования (САПР). Обзор САПР для различных уровней проектирования. Языки описания аппаратуры HDL. Сквозное проектирование цифровых устройств на основе ПЛИС в САПР ISE WebPACK Xilinx		1		2/2И	25	Самостоятельное изучение учебной литературы. Подготовка к практическим занятиям	Контрольная работа	ПК-1.1, ПК-1.2
Итого по разделу		2		2/2И	45			
2. Моделирование								
2.1 Виды моделирования и типы моделей на различных этапах проектирования. Использование VHLD- и SPICE-моделей. Моделирование работы цифровых устройств с помощью встроенного в САПР ISE WebPACK симулятора Isim.	5	1		2/2И	25	Самостоятельное изучение учебной литературы. Подготовка к практическим занятиям	Контрольная работа	ПК-1.1, ПК-1.2

2.2 Разработка проектной документации. Конфигурирование ПЛИС с помощью встроенной в САПР ISE WebPACK программы Impact. Тестирование готовых устройств. JTAG-интерфейс.		1		2	23,4	Самостоятельное изучение учебной литературы. Подготовка к практическим занятиям	Контрольная работа	ПК-1.1, ПК-1.2
Итого по разделу		2		4/2И	48,4			
Итого за семестр		4		6/4И	93,4		зачёт	
Итого по дисциплине		4		6/4И	93,4		зачет	

5 Образовательные технологии

Лекционные занятия по дисциплине «Основы проектирования электронной компонентной базы» проводятся в традиционной форме с использованием мультимедийного оборудования. Практические занятия проходят как в традиционной форме, так и в интерактивной форме, где студентам заранее предлагается ознакомиться с информацией по теме занятия для подготовки вопросов преподавателю, таким образом, практическое занятие проходит по типу «вопросы–ответы–дискуссия». На всех практических занятиях также применяются элементы занятия-визуализации, за счет представления части материала с помощью заранее подготовленных презентаций, слайдов с помощью мультимедийного оборудования.

Теоретический материал, освоенный студентами самостоятельно, закрепляется на практических занятиях, на которых выполняются индивидуальные и групповые задания по пройденной теме. Часть практических занятий проводится в виде традиционных семинаров с целью более глубокого и полного усвоения теоретического материала по данной теме. Для этого студентам предлагается готовить доклады по рассматриваемой теме с дальнейшим обсуждением в ходе практического занятия (учебных дискуссий). На практических занятиях также применяются метод контекстного обучения, работы в команде и метод case-study, позволяющие усвоить учебный материал путём вы-явления связей между конкретным знанием и его применением, а также анализа конкретных ситуаций и поиска решений в группе студентов. Защита результатов практических заданий проходит в виде диалога преподавателя и студента, преподавателем задаются контрольные вопросы с целью выяснения глубины знаний студента по дан-ному разделу, при этом пробелы в знаниях студента восполняются дополнительными пояснениями, комментариями преподавателя.

В качестве оценочных средств на протяжении семестра используются контрольные работы. Самостоятельная работа студентов заключается в проработке материала при подготовке к практическим занятиям.

6 Учебно-методическое обеспечение самостоятельной работы обучающихся

Представлено в приложении 1.

7 Оценочные средства для проведения промежуточной аттестации

Представлены в приложении 2.

8 Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) Основная литература:

1. Мурсаев, А. Х. Практикум по проектированию на языках VerilogHDL и SystemVerilog : учебное пособие / А. Х. Мурсаев, О. И. Буренева. — 2-е изд., стер. — Санкт-Петербург : Лань, 2018. — 120 с. — ISBN 978-5-8114-2560-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/103142 (дата обращения: 20.10.2020). — Режим доступа: для авториз. пользователей.

2. Петров, М. Н. Моделирование компонентов и элементов интегральных схем : учебное пособие / М. Н. Петров, Г. В. Гудков. — Санкт-Петербург : Лань, 2011. — 464 с. — ISBN 978-5-8114-1075-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/661 (дата обращения: 20.10.2020). — Режим доступа: для авториз. пользователей.

б) Дополнительная литература:

1. Щука, А.А. Электроника [Текст] / А.А. Щука. - СПб.: БВХ Петербург, 2008. – 751с. Режим доступа: http://znanium.com/catalog/product/350420 .

в) Методические указания:

Методические рекомендации по выполнению практических заданий представлены в приложении 3.

г) Программное обеспечение и Интернет-ресурсы:

Программное обеспечение

Наименование ПО	№ договора	Срок действия лицензии
MS Windows 7 Professional(для классов)	Д-1227-18 от 08.10.2018	11.10.2021
MS Office 2007 Professional	№ 135 от 17.09.2007	бессрочно
ISE WebPACK Xilinx	Бесплатное ПО	бессрочно

Профессиональные базы данных и информационные справочные системы

Название курса	Ссылка
Поисковая система Академия Google (Google Scholar)	URL: https://scholar.google.ru/

9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

Лекционная аудитория: Мультимедийные средства хранения, передачи и представления информации (ауд. 458).

Компьютерный класс: Персональные компьютеры с установленным специализированном ПО для проектирования элементной базы (ISE WebPACK Xilinx) (ауд. 357).

Аудитория для самостоятельной работы: компьютерный класс Персональные компьютеры с установленным специализированном ПО для проектирования элементной базы (ISE WebPACK Xilinx) (ауд. 458).

Помещение для хранения и профилактического обслуживания учебного оборудования: Стеллажи, сейфы для хранения учебного оборудования. Инструменты для ремонта оборудования.

Учебно-методическое обеспечение самостоятельной работы обучающихся

По дисциплине «Основы проектирования электронной компонентной базы» предусмотрена аудиторная и внеаудиторная самостоятельная работа обучающихся.

Аудиторная самостоятельная работа студентов предполагает решение контрольных задач на практических занятиях.

Примерные аудиторные контрольные работы (АКР):

АКР №1 «Современная электронная компонентная база. Классификация. Область применения»:

1. Классификация электровакуумных электронных приборов.
2. Сфера применения электровакуумных приборов.
3. Перспективы применения электровакуумных приборов.
4. Какие элементы входят в модель прибора вакуумной электроники.
5. Какие электронно-лучевые приборы вы знаете?
6. Что такое полупроводниковый электронный прибор?
7. Классификация полупроводниковых приборов.
8. Сфера применения полупроводниковых приборов.
9. Какие полупроводниковые приборы вы знаете?
10. Перспективы применения полупроводниковых приборов.
11. Элементная база оптоэлектроники.
12. Сфера применения оптоэлектроники
13. Что такое интегральная схема?
14. Классификация интегральных схем?
15. Что значит технологическая норма интегральной схемы?
16. Элементная база интегральных схем.
17. Что такое заказная ИС
18. Что такое полузаказная ИС?
19. Что представляет собой базовый матричный кристалл?
20. Что такое программируемая логическая интегральная схема?
21. Что такое «система на кристалле»?
22. Что такое логический элемент ИС?
23. Что такое логическая ИС комбинационного типа?
24. Что такое логическая ИС последовательностного типа?
25. Какие типы логических ячеек (логики) вы знаете?

АКР №2 «Проектирование электронной компонентной базы: основные этапы и уровни проектирования»:

1. Перечислите основные этапы производства ИС
2. Что включает в себя спецификация на разрабатываемую ИС
3. Какова иерархия проектирования СБИС.
4. Что такое кремниевый уровень проектирования. Какие примитивы применяются на данном уровне.
5. Что такое транзисторный уровень проектирования. Какие примитивы применяются на данном уровне.
6. Что такое вентильный уровень проектирования. Какие примитивы применяются на данном уровне.
7. Что такое регистровый уровень проектирования. Какие примитивы применяются на данном уровне.
8. Что такое процессорный уровень проектирования. Какие примитивы применяются на данном уровне.
9. Что такое системный уровень проектирования. Какие примитивы применяются на данном уровне.

10. В чём заключается принцип управления сложностью (абстрагирование) при разработке электроники.
11. Какова современная инфраструктура производства ИС.
12. Что представляют собой кремниевые фабрики.
13. Что такое IP-блок.
14. Классификация IP-блоков
15. В чём отличие программных IP-блоков от аппаратных IP-блоков.
16. Что представляют собой топологические IP-блоки.
17. Этапы проектирования заказной ИС.
18. Этапы проектирования ИС на стандартных ячейках.
19. Этапы проектирования схемы на базе ПЛИС.

АКР №3 «Системы автоматизированного проектирования (**САПР**). Обзор САПР для различных уровней проектирования. Языки описания аппаратуры **HDL**. Сквозное проектирование цифровых устройств на основе **ПЛИС** в САПР **ISE WebPACK Xilinx**»:

1. Что такое язык описания аппаратуры HDL.
2. Каковы преимущества разработки схемы на базе HDL по сравнению со схемотехническим способом.
3. Что такое логический синтез схемы.
4. Какие САПР разработки ИС вы знаете?
5. Какие САПР для разработки схем на базе ПЛИС вы знаете?
6. Логический синтез ИС на стандартных ячейках.
7. Логический синтез схем на ПЛИС.
8. Что такое критический путь цифровой схемы?
9. Какие языки описания аппаратуры вы знаете?
10. Чем отличаются синтезируемые структуры языка HDL от несинтезируемых?
11. Какими способами можно повысить быстродействие цифровой схемы?
12. В чём заключается компромисс площадь кристалла/быстродействие?
13. Что такое синхронная цифровая схема?
14. Что включает в себя описание интерфейса (entity declaration) в языке VHDL.
15. Какие значения поддерживает тип данных **STD_LOGIC** в языке VHDL.
16. Чем отличается тип данных **STD_LOGIC** от **STD_LOGIC_VECTOR** в языке VHDL.
17. В чём отличие объекта **port** от объекта **signal** в языке VHDL.
18. Какие значения поддерживает тип данных **unsigned** в языке VHDL. Является ли данный тип синтезируемым?
19. Какие значения поддерживает тип данных **integer** в языке VHDL. Является ли данный тип синтезируемым?
20. Является ли высокоимпедансное состояние **Z** синтезируемым в языке VHDL? Если да, то какая схема синтезируется?
21. Каков синтаксис структуры «присваивание по условию» (**conditional signal assignment**) в языке VHDL. В Какую схему данная структура синтезируется?
22. Каков синтаксис структуры «Присваивание по выбору» (**selected signal assignment**) в языке VHDL. В Какую схему данная структура синтезируется?
23. Каков синтаксис оператора **if** в языке VHDL. В Какую схему данная структура синтезируется?
24. Каков синтаксис оператора **case** в языке VHDL. В Какую схему данная структура синтезируется?
25. Для чего в языке VHDL используется структура **process**?
26. Разработать одноразрядную схему сравнения на вентильном уровне на языке VHDL.
27. Разработать на языке VHDL схему дешифратора 2 в 4.

28. Разработать на языке VHDL схему преобразователя двоичного кода в семисегментный.
29. Разработать модуль на VHDL, вычисляющий четырехвходовую функцию XOR (исключающее ИЛИ).
30. Разработать на языке VHDL схему 4-х разрядного счётчика.
31. Разработать на языке VHDL схему 4-х разрядного сумматора чисел со знаком.
32. Для чего применяется generic в языке VHDL.
33. Что значит параметризованная схема?
34. Разработать на языке VHDL схему 8-и разрядного регистра.
35. Разработать на языке VHDL схему сдвигового регистра с параллельной загрузкой.
36. Разработать на языке VHDL схему конечного автомата для детектирования переднего фронта сигнала.

АКР №3 «Виды моделирования и типы моделей на различных этапах проектирования. Использование **VHLD**- и **SPICE**-моделей. Моделирование работы цифровых устройств с помощью встроенного в САПР **ISE WebPACK** симулятора **Isim**»:

1. Что такое среда тестирования (**testbench**).
2. Что такое среда тестирования с самопроверкой?
3. Что такое функциональное моделирование?
4. Уровни верификации СБИС.
5. Назначение функциональных блоков stimulus, checker и monitor в среде тестирования?
6. Что такое рандомизация тестовых воздействий (stimulus)
7. Что такое ограниченная (constrained) рандомизация тестовых воздействий
8. Чем отличается кодовое покрытие от функционального покрытия при верификации схемы?
9. Что такое метрика при верификации схемы? Какие метрики используются?
10. Что такое эмуляция схемы?
11. Что такое формальная проверка эквивалентности на этапе синтезирования схемы?
12. Что такое временное моделирование схемы?
13. Написать testbench на VHDL с самопроверкой для одноразрядной схемы сравнения.
14. Написать testbench на VHDL для схемы дешифратора 2 в 4.
15. Написать testbench на VHDL для схемы преобразователя двоичного кода в семисегментный.
16. Написать testbench на VHDL с самопроверкой для 4-х разрядной схемы XOR (исключающее ИЛИ).
17. Написать testbench на VHDL для схемы 4-х разрядного счётчика.
18. Написать testbench на VHDL для схемы 4-х разрядного сумматора чисел со знаком.
19. Написать testbench на VHDL для схемы 8-и разрядного регистра.
20. Написать testbench на VHDL для схемы сдвигового регистра с параллельной загрузкой.
21. Написать testbench на VHDL для схемы детектирования переднего фронта сигнала.

АКР №5 «Разработка проектной документации. Конфигурирование ПЛИС с помощью встроенной в САПР **ISE WebPACK** программы **Impact**. Тестирование готовых устройств. JTAG-интерфейс»:

1. Что такое имплементация?

2. Каковы основные рекомендации по размещению блоков на кристалле?
3. Что такое слой металлизации в ИС?
4. Что такое трассировка?
5. Применение манхэтоновской геометрии при трассировке.
6. Что такое физическая верификация ИС?
7. Что такое DRC проверка?
8. Что такое ERC проверка?
9. Что такое LVS проверка?
10. Что такое GDSII формат?
11. Что такое JTAG тестирование?
12. Что такое цепочка тестирования?
13. Что такое встроенное самотестирование BIST?
14. Встроенное самотестирование памяти.
15. Что такое периферийное сканирование?
16. Реализовать одноразрядную схему сравнения на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
17. Реализовать схему дешифратора 2 в 4 на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
18. Реализовать схему преобразователя двоичного кода в семисегментный на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
19. Реализовать четырехвходовую функцию XOR (исключающее ИЛИ) на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
20. Реализовать схему 4-х разрядного счётчика на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
21. Реализовать схему 4-х разрядного сумматора чисел со знаком на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
22. Реализовать схему 8-и разрядного регистра на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
23. Реализовать схему сдвигового регистра с параллельной загрузкой на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.
24. Реализовать схему детектирования переднего фронта сигнала на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.

Внеаудиторная самостоятельная работа обучающихся осуществляется в виде изучения литературы по соответствующему разделу с проработкой материала.

Оценочные средства для проведения промежуточной аттестации

а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
ПК-1: Способен разрабатывать структурные и функциональные схемы электронных систем и комплексов, принципиальных схем устройств с использованием средств компьютерного проектирования, проведением проектных расчетов и технико-экономическим обоснованием принимаемых решений		
ПК-1.1	Разрабатывает эскизный проект, включающий: выбор структурной схемы электронного устройства или системы путем сопоставления различных вариантов и их оценки с точки зрения технических и экономических требований; рассчитывает все необходимые показатели структурной схемы электронного устройства или системы, в том числе показатели качества; выбирает и обосновывает схемы вспомогательных устройств	– *Что такое язык описания аппаратуры HDL.* – *Каковы преимущества разработки схемы на базе HDL по сравнению со схемотехническим способом.* – *Что такое логический синтез схемы.* – *Какие САПР разработки ИС вы знаете?* – *Какие САПР для разработки схем на базе ПЛИС вы знаете?* – *Логический синтез ИС на стандартных ячейках.* – *Логический синтез схем на ПЛИС.* – *Что такое критический путь цифровой схемы?* – *Какие языки описания аппаратуры вы знаете?* – *Чем отличаются синтезируемые структуры языка HDL от несинтезируемых?* – *Какими способами можно повысить быстродействие цифровой схемы?* – *В чём заключается компромисс площадь кристалла/быстродействие?* – *Что такое синхронная цифровая схема?*
		Практические задания: – *Разработать одноразрядную схему сравнения на вентильном уровне на языке VHDL.* – *Разработать на языке VHDL схему дешифратора 2 в 4.* – *Разработать на языке VHDL схему преобразователя двоичного кода в семисегментный.* – *Разработать модуль на VHDL, вычисляющий четырехвходовую функцию XOR (исключающее ИЛИ).* – *Разработать на языке VHDL схему 4-х разрядного счётчика.* – *Разработать на языке VHDL схему 4-х разрядного сумматора чисел со знаком.* – *Разработать на языке VHDL схему 8-и разрядного регистра.* – *Разработать на языке VHDL схему*

		сдвигового регистра с параллельной загрузкой. – *Разработать на языке VHDL схему конечного автомата для детектирования переднего фронта сигнала.* – *Реализовать одноразрядную схему сравнения на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему дешифратора 2 в 4 на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему преобразователя двоичного кода в семисегментный на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать четырехвходовую функцию XOR (исключающее ИЛИ) на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему 4-х разрядного счётчика на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему 4-х разрядного сумматора чисел со знаком на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему 8-и разрядного регистра на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему сдвигового регистра с параллельной загрузкой на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.* – *Реализовать схему детектирования переднего фронта сигнала на базе ПЛИС Spartan 3E. Определить быстродействие схемы и затраченные ресурсы ПЛИС.*
ПК-1.2	Производит технико-экономическое обоснование принятого решения с расчетами себестоимости устройства и стоимости его эксплуатации; сравнивает с	– *Перечислите основные этапы производства ИС* – *Что включает в себя спецификация на разрабатываемую ИС* – *Какова иерархия проектирования СБИС.* – *Что такое кремниевый уровень проектирования. Какие*

	аналогами по технико-экономическим характеристикам	*примитивы применяются на данном уровне.* – *Что такое транзисторный уровень проектирования. Какие примитивы применяются на данном уровне.* – *Что такое вентильный уровень проектирования. Какие примитивы применяются на данном уровне.* – *Что такое регистровый уровень проектирования. Какие примитивы применяются на данном уровне.* – *Что такое процессорный уровень проектирования. Какие примитивы применяются на данном уровне.* – *Что такое системный уровень проектирования. Какие примитивы применяются на данном уровне.* – *В чём заключается принцип управления сложностью (абстрагирование) при разработке электроники.* – *Какова современная инфраструктура производства ИС.* – *Что такое IP-блок.* – *Классификация IP-блоков* – *Что представляют собой топологические IP-блоки.* – *Этапы проектирования заказной ИС.* – *Этапы проектирования ИС на стандартных ячейках.* – *Этапы проектирования схемы на базе ПЛИС.*
		Практические задания: – *Разработать двухразрядную схему сравнения на основе двух экземпляров одноразрядной схемы сравнения. Использовать комментарии для описания кода.* – *Разработать на языке VHDL схему дешифратора 3 в 8 на основе экземпляров схемы дешифратора 2 в 4. Использовать комментарии для описания кода.* – *Разработать на языке VHDL схему 16-и разрядного сумматора чисел со знаком на основе экземпляров 4-х разрядного сумматора. Использовать комментарии для описания кода.* – *Разработать на языке VHDL схему 8-и разрядного регистра. Использовать комментарии для описания кода.* – *Разработать на языке VHDL схему*

		конечного автомата для реализации защиты от дребезга. Использовать комментарии для описания кода. *Подготовить проектную документацию: RTL-код и файл ограничений (топологических и временных) для реализации проекта на базе ПЛИС для следующих проектов:* *– Восьмиразрядная схема сдвига с управляющим входом, определяющим направление сдвига.* *– Приоритетный шифратор 8 в 3* *– Преобразователь двоичного кода в двоично-десятичный* *– 4-х разрядный сумматор чисел с плавающей точкой.* *– 8-и разрядный FIFO буфер* *– 4-х разрядный ШИМ* *– Сторожевой таймер* *– Схема стека* *– Арифметико-логическое устройство* *– Регистровый файл* *– Схема деления*

б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:

Промежуточная аттестация по дисциплине «Основы проектирования электронной компонентной базы» включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме зачёта.

Показатели и критерии оценивания зачёта:

- на оценку «**зачтено»** – обучающийся демонстрирует высокий или средний уровень сформированности компетенций: основные знания, умения освоены, но допускаются незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.
- на оценку **«не зачтено»** – обучающийся демонстрирует знания не более 20% теоретического материала, допускает существенные ошибки, не может показать интеллектуальные навыки решения простых задач.

Методические указания

Введение

Традиционное схемное проектирование достигло в своем развитии предела и используется в настоящее время лишь для сравнительно простых проектов разрабатываемой электроники. Основным недостатком схемного описания проектов сложной электроники, включающей большое количество компонентов и модулей, является высокая трудоемкость отладки. На смену схемному описанию пришли языки описания аппаратуры, такие как VHLD и Verilog. В отличие от традиционных языков программирования, описывающих последовательность выполняемых команд, языки описания аппаратуры описывают структуру и связи разрабатываемого устройства. Такое описание разрабатываемого устройства называется структурным (параллельным). Второй подход к описанию устройства называется поведенческим (последовательным). При таком подходе описывается не структура устройства, а порядок его функционирования (поведение). Языки описания аппаратуры включат как параллельные, так и последовательные функции, т.е. с их помощью можно описать как структуру разрабатываемого устройства, так и его поведение в зависимости от входных воздействий. Здесь следует сделать важное замечание:

Последовательное (поведенческое) описание устройства не всегда может быть корректно скомпилировано в синтезируемые конструкции, т.е. синтезированная схема устройства может работать не так, как задумывал разработчик. В тоже время структурное (параллельное) описание однозначно синтезируется в описанную структуру. Таким образом, при описании разрабатываемого устройства следует отдавать предпочтение структурному описанию или, как говорят, использовать синтезируемые конструкции. Последовательное (поведенческое) описание в свою очередь эффективно используется на этапе функционального моделирования работы разрабатываемого устройства. Тем не менее, использование последовательных операторов языков описания аппаратуры для синтеза разрабатываемого устройства допускается при условии жесткого соблюдения ряда правил (данные правила будут рассмотрены в соответствующей лабораторной работе).

Данные методичесские призван научить основам проектирования электроники с помощью языка описания аппаратуры VHLD. При этом основной упор делается на развитие навыков написания грамотного и самое главное синтезируемого кода, т.е. такого кода, который будет гарантированно скомпилирован в работающую схему. Поэтому вместо подробного описания синтаксиса VHLD и всевозможных конструкций языка практикум ограничивается обзором сравнительно небольшого набора синтезируемых конструкций, использующих десяток типовых шаблонов VHDL-кода, синтезируемых в типовые схемы. Данные шаблоны могут быть легко интегрированы при проектировании больших сложных систем.

Для обеспечения процесса сквозного проектирования вплоть до готового устройства было решено использовать программируемую логику (ПЛИС), а именно отладочную плату **NI Digital Electronics FPGA Board** на основе ПЛИС **Xilinx Spartan-3E**. В качестве программного обеспечения используется САПР **ICE WebPACK** фирмы Xilinx.

Описание платы NI Digital Electronics FPGA

Плата **NI Digital Electronics FPGA** является результатом совместного сотрудничества компаний NI и Xilinx, являющихся крупнейшими в мире производителями программного обеспечения измерительных систем и программируемых логических интегральных схем (ПЛИС). Внешний вид платы NI Digital Electronics FPGA представлен на рисунке 1. NI Digital Electronics FPGA Board содержит шесть каналов для ввода аналоговых входных сигналов AI0 – AI5, а также 32 цифровые входные (выходные) линии общего пользования – GPIO31.

Описание сигналов, подаваемых на сигнальные зоны макетирования: BB1 – зона макетирования для подключения к ЦАП, АЦП, кнопкам, движковым переключателям, внешнему синхросигналу и линиям общего назначения ПЛИС. BB2 - зона макетирования для подключения к линиям общего назначения ПЛИС и источникам питания. BB3 - зона макетирования для управления семисегментным индикатором на два знакоместа, светодиодами, подключения к источникам питания BB4 - зона макетирования для подключения к сигналам NI ELVIS, включая аналоговые входные сигналы, аналоговые выходные сигналы, функциональные генераторы, источники питания, цифровые входные/выходные сигналы. BB5 – зона макетирования для подключения к сигналам NI ELVIS, включая регулируемые источники питания, цифровые входные/выходные сигналы, выходные сигналы счѐтчиков, общие точки сигналов.

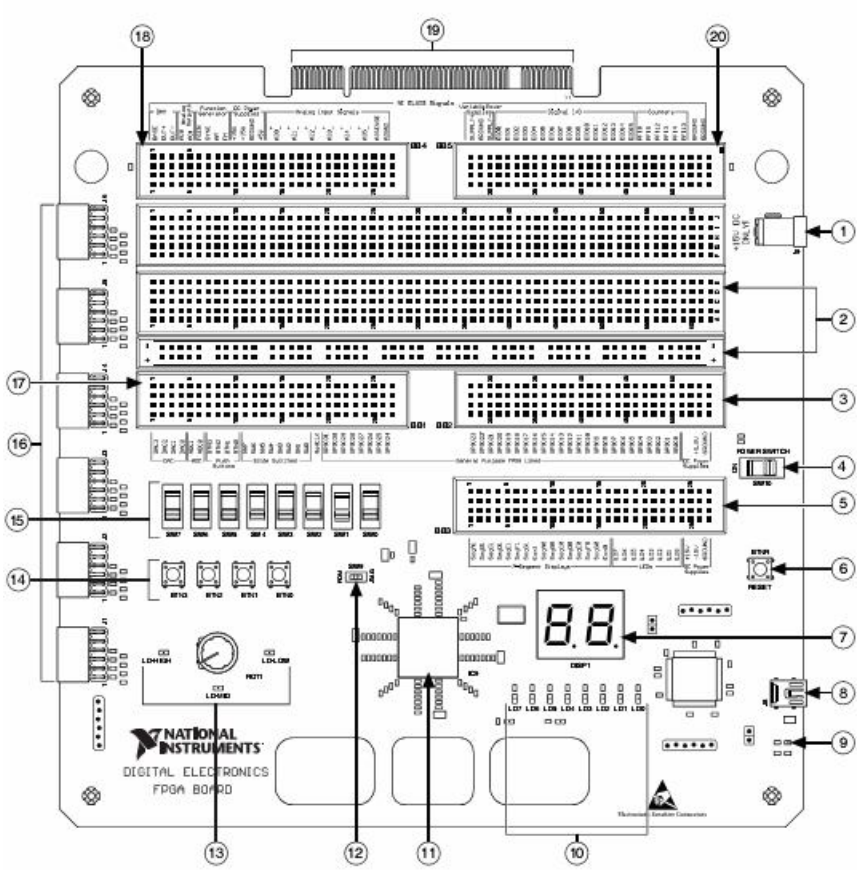

Рисунок 1 – Внешний вид платы NI Digital Electronics FPGA Board:

1– разъем питания (15В) ; 2–макетной платы для возможности создания дополнительной обвязки; 3– блок макетной платы ВВ2; 4– выключатель питания; 5– блок макетной платы ВВ3; 6– кнопка сброса; 7– семисегментные индикаторы; 8– USB-соединитель; 9– LD-G- светодиод; 10– светодиоды; 11– FPGA Xilinx Spartan 3E; 12– переключатель прошивки ПЛИС через ROM/JTAG ; 13– датчик угла поворота с кнопкой; 14– кнопки; 15– движковые переключатели; 16– 6 коннекторов типа PMOD (2x6); 17– блок макетной платы ВВ1; 18– блок макетной платы ВВ4; 19– (NI ELVIS)- соединитель; 20– блок макетной платы ВВ5; 21 – программатор; 22– кварцевый генератор 50МГц; 23– защита платы от статического электричества

Краткое описание САПР WebPack

WebPACK – это САПР проектирования цифровых устройств на базе микросхем ПЛИС CPLD и FPGA фирмы Xilinx. Данная система является бесплатным вариантом коммерческой САПР этой же фирмы под названием ISE и доступна для свободного скачивания через сеть Internet (www.xilinx.com). Основное отличие бесплатной версии от ее платного аналога состоит в отсутствии поддержки микросхем, емкость которых выше 1,5 млн системных вентилей.

WebPACK состоит из набора модулей, каждый из которых выполняет свои специализированные функции. Основные модули пакета следующие:

- редактор схемного ввода;

- текстовый редактор с поддержкой языков описания аппаратуры VHDL и Verilog;

- CORE Generator – генератор оптимизированных IP-ядер;

- редактор тестовых воздействий для программы моделирования;

- программа функционального и временного моделирования;

- генератор VHDL/Verilog кода;

- программа автоматического размещения и трассировки ПЛИС;—программы «ручного» размещения и оптимизации проекта;

- программа загрузки конфигурационной последовательности в ПЛИС FPGA и программирования ПЛИС CPLD и ППЗУ.

Большинство модулей САПР WebPACK имеют как графический интерфейс пользователя, так и интерфейс командной строки. САПР WebPACK может работать под операционными системами Windows, Linux и Sun Solaris.

Процесс разработки цифровых устройств в среде WebPACK состоит из следующих этапов.

1. Ввод описания проектируемого устройства в схемотехнической форме или с использованием языков описания аппаратуры (HDL), таких, как VHDL и Verilog.

2. Синтез устройства, то есть преобразование описания устройства, полученного на первом этапе, в описание на уровне логических вентилей.

3. Реализация устройства, то есть преобразование описания устройства на уровне логических вентилей в физическое описание для конкретной микросхемы ПЛИС.

4. Формирование конфигурационной последовательности для микросхе-

мы ПЛИС.

После каждого из этапов 1,2 и 3 возможно, а в большинстве случаев и необходимо для успешного завершения проекта выполнение процедуры моделирования и верификации полученного описания устройства.

На рисунке 2 представлена обобщенная схема проектирования цифровых устройств в САПР WebPACK.

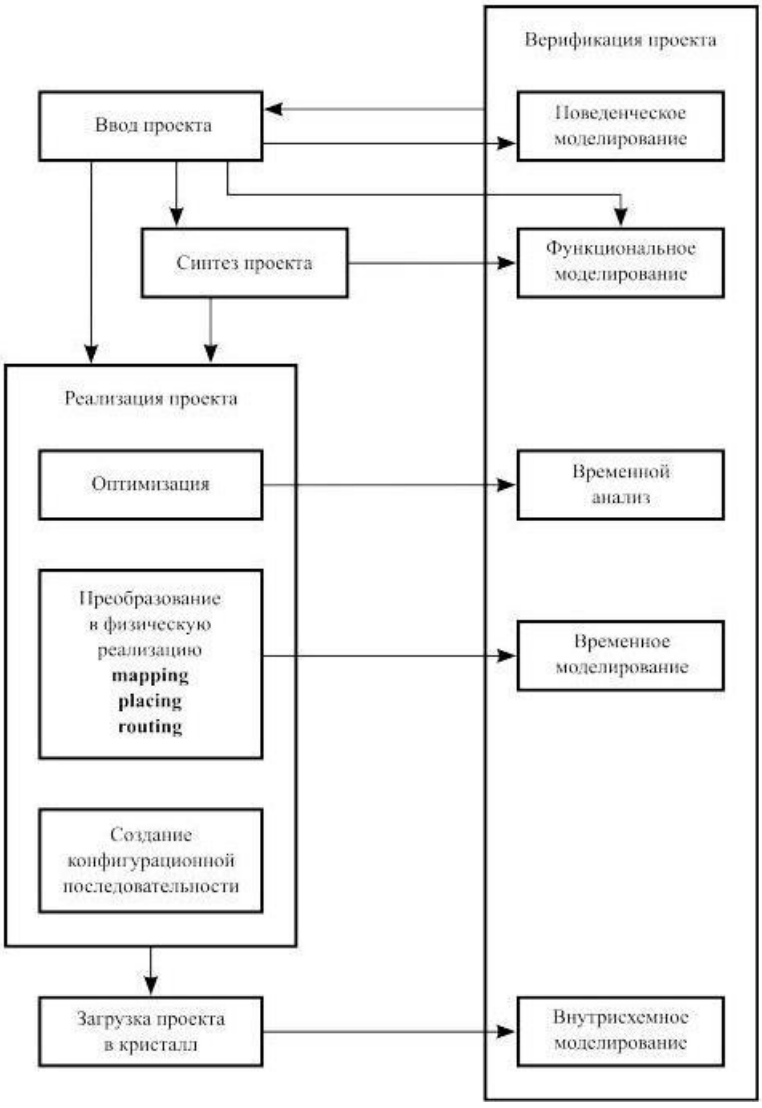

Рисунок 2 - Обобщенная схема проектирования цифровых устройств в САПР WebPACK

Проектирование комбинационных схем на вентильном уровне

Введение

В данной главе на примере простой схемы сравнения подробно рассмотрена структура синтезируемого VHDL-кода. Описание работы цифровых комбинационных схем в данной работе представлено исключительно на вентильном уровне абстракции (gate level), т.е. применяются только базовые логические вентили, реализующие основные логические операции (НЕ, И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ). Рассмотрен структурный (иерархический) подход к проектированию цифровых систем, при котором система на верхнем уровне проектирования рассматривается как соединение более простых унифицированных блоков, которые, в свою очередь, также строятся из более простых блоков и т.д.

1.1 Описание схем в дизъюнктивной нормальной форме

Рассмотрим одноразрядную схему сравнения (1-bit equality comparator) , содержащую два входа *i0*, *i1* и один выход *eq*. Выход *eq* схемы сравнения принимает активное значение при условии, что входы *i0* и *i1* эквивалентны. Таблица истинности (truth table) одноразрядной схемы сравнения представлена в таблице 1.1.

Таблица 1.1 – Таблица истинности одноразрядной схемы сравнения

i0	i1	eq
0	0	1
0	1	0
1	0	0
1	1	1

Допустим, что для реализации одноразрядной схемы сравнения мы решили использовать базовые логические вентили (logic gates), такие как: *НЕ* (*NOT*), *И* (*AND*), *ИЛИ* (*OR*), *Исключающее ИЛИ* (*XOR*). Одной из форм записи логической функции, описывающей работу схемы, является запись функции в совершенной дизъюнктивной нормальной форме, т.е. в виде суммы минтермов (sum-of-products canonical form) . Логическая функция одноразрядной схемы сравнения, записанная в совершенной дизъюнктивной нормальной форме, имеет вид:

$$eq = i0 \cdot i1 + \overline{i0} \cdot \overline{i1}$$

Один из возможных вариантов VHDL-кода, описывающего рассматриваемую схему сравнения приведен в листинге 1.1. Проанализируем языковые конструкции приведенного кода в следующих подразделах.

Листинг 1 – Реализация одноразрядной схемы сравнения на вентильном уровне

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
---------------------------------
entity eq1 is
    Port ( i0 : in  STD_LOGIC;
           i1 : in  STD_LOGIC;
           eq : out  STD_LOGIC);
end eq1;
---------------------------------
```

```
architecture sop_arch of eq1 is
       signal p0 , p1 : STD_LOGIC;
begin
       eq <= p0 or p1 ;                    -- сумма минтермов
       p0 <= (not i0) and (not i1) ;       -- определение минтерма
       p1 <= i0 and i1 ;                   -- определение минтерма
end sop_arch;
```

1.1.1 Основные лексические правила

Код VHDL является нечувствительным к способу написания (регистру) символов (case insensitive), т.е. прописные и строчные буквы являются взаимозаменяемыми. Кроме того в коде VHDL допускается вставлять дополнительные пробелы и пустые строки. Использование дополнительных пробелов и пустых строк позволяет сделать код более понятным.

Идентификатором в языке VHDL называется имя объекта (константы, переменной, функции, сигнала, порта, подпрограммы, объекта проекта, метки). Идентификатор может содержать до 26 букв, цифр и нижних подчеркиваний (не допускается использовать более одного символа нижнего подчеркивания подряд). Начинаться идентификатор должен только с буквы. В примере (листинг 1.1) используются следующие идентификаторы: eq1, i0, i1, eq, sop_arch, p0, p1.

В языке VHDL существуют зарезервированные ключевые слова. В данном учебном пособии все ключевые слова выделены синим цветом (листинг 1.1).

Комментарий начинается с двух смежных дефисов и продолжается до конца строки. При синтезе VHDL-проекта комментарии игнорируются. В данном учебном пособии все комментарии выделены зеленым цветом (листинг 1.1).

1.1.2 Используемые библиотеки

Описание объекта на VHDL состоит из трех частей (листинг 1.1): объявления используемых библиотек (library, use), описания интерфейса объекта (entity) и его внутренней структуры (architecture).

Первые две строчки в примере (листинг 1.1) объявляют используемые библиотеки:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
```

В данном случае объявляется библиотека IEEE.STD_LOGIC_1164, что позволяет использовать соответствующие типы данных, операторы и функции, определенные в данной библиотеке.

1.1.3 Интерфейс объекта

Описание интерфейса (entity declaration) включает в себя объявление имени объекта и объявление входных и выходных портов объекта. Ниже приведено описание интерфейса из примера (листинг 1.1):

```
entity eq1 is                          -- декларация имени объекта
```

```
    Port ( i0 , i1 : in  STD_LOGIC;      -- декларация входных портов
           eq : out  STD_LOGIC);         -- декларация выходного порта
end eq1;
```

В данном примере в первой строке объявляется имя объекта eq1. Структура port описывает входные и выходные порты в следующем формате:

имя_порта1, имя_порта2, . . . : режим_сигнала тип_сигнала ;

Для входных портов указывается входной режим сигнала (in), для выходных портов указывается выходной режим сигнала (out). Также предусмотрен режим inout для двунаправленных портов.

1.1.4 Тип данных и операторы

VHDL – язык со строгой типизацией. Это значит, что данные определенного типа могут принимать значения, соответствующие только данному типу (например, логический тип данных BOOLEAN может принимать только значения *ИСТИНА* (*TRUE*) и *ЛОЖНО* (*FALSE*) и никакие другие) и с данными определенного типа могут выполняться операции, соответствующие только данному типу (например, для данных типа BOOLEAN используются только логические операторы). Несмотря на то, что в языке VHDL применяется множество типов данных, в данном учебном пособии используются далеко не все – в основном применяется тип STD_LOGIC и его варианты. Это связано с тем, что основной упор в данном учебном пособии сделан на синтезируемые конструкции языка VHDL, т.е. такие конструкции, на основе которых может быть синтезирована реально работающая схема.

Тип данных STD_LOGIC определен в библиотеке IEEE.STD_LOGIC_1164 и содержит девять возможных значений. Три значения: ‘0’, ’1’ и ’Z’, соответствующие логическому нулю, логической единице и состоянию высокого омического сопротивления (Z-состоянию), могут быть синтезированы. Два других значения ‘U’ и ’X’, которые означают “uninitialize” (“неинициализированный”) и “unknown” (“неизвестный”), могут встретиться при симуляции проекта (например, когда сигналы ‘0’ и ‘1’ соединяются в одной точке). Оставшиеся четыре значения ‘-’, ’H’, ’L’ и ’W’ не используются в данном учебном пособии.

Цифровой сигнал зачастую представляет собой двоичной слово, т.е. набор битов. В данном случае, такой сигнал удобно представлять типом данных STD_LOGIC_VECTOR. Например, 8-ми битный входной порт данных можно объявить следующим образом:

```
a: in  STD_LOGIC_VECTOR (7 downto 0);
```

Запись “(7 downto 0)” означает, что в формате числа старший значащий бит (MSB) располагается крайним слева, а младший значащий бит (LSB) – крайним справа (такая форма записи является общепринятой в электронике, поэтому в данном учебном пособии используется именно этот формат). Для выделения части слова, например старших четырех бит в указанном примере, можно использовать запись “a(7 downto 4)”; для выделения отдельного бита из слова, например нулевого бита в указанном примере, применяется запись “a(0)”.

Для данных типа STD_LOGIC и STD_LOGIC_VECTOR определен ряд логических операторов, таких как: *НЕ* (*NOT*), *И* (*AND*), *ИЛИ* (*OR*), *Исключающее ИЛИ* (*XOR*). Для цифровых слов типа STD_LOGIC_VECTOR указанные логические операции выполняются побитно. В языке VHDL указанные логические операторы обладают одинаковым приоритетом, поэтому для

определения порядка выполнения логического выражения необходимо применять скобки, например:

```
(a and b) or (c and d)
```

1.1.5 Структура объекта

Внутренняя структура объекта (architecture body) описывает работу схемы (листинг 1.1):

```
architecture sop_arch of eq1 is

    signal p0 , p1 : STD_LOGIC;

begin

    eq <= p0 or p1 ;                    -- сумма минтермов

    p0 <= (not i0) and (not i1) ;       -- определение минтерма

    p1 <= i0 and i1 ;                   -- определение минтерма

end sop_arch;
```

VHDL позволяет множеству различных структур объекта (architecture body) привязываться к интерфейсу объекта (entity), поэтому структура объекта должна иметь уникальное имя для своей идентификации, например “sop_arch” в примере выше (“sum_of_products architecture”).

Структура объекта (architecture body) может включать в себя секцию для объявления внутренних переменных (сигналов) объекта, вспомогательных констант и т.д. В приведенном примере, дополнительно объявляются две внутренние переменные:

```
    signal p0 , p1 : STD_LOGIC;
```

Основная часть описания структуры объекта (architecture body) помещена между ключевыми словами begin и end. В приведенном примере структура объекта описывается тремя выражениями (statement):

```
    eq <= p0 or p1 ;

    p0 <= (not i0) and (not i1) ;

    p1 <= i0 and i1 ;
```

При этом, в отличие от стандартных языков программирования (например *C*), где выражения (строки программы) выполняются последовательно, в языках описания аппаратуры HDL приведенные выражения (statement) описывают части реализуемой схемы и потому выполняются параллельно (concurrent statements). Переменная слева, при этом, рассматривается как выходной сигнал схемы, выражение справа определяет функцию работы схемы и соответствующие входные сигналы. В качестве примера рассмотрим следующее выражение:

```
    eq <= p0 or p1 ;
```

Данное выражение описывает схему, реализующую операцию ИЛИ между входными сигналами *p0* и *p1*. При изменении значений сигналов *p0* и *p1* выражение пересчитывается. Результат расчета присваивается выходному сигналу *eq* через определенную временную задержку, заданную по умолчанию.

Графическое представление объекта (листинг 1.1) представлено на рисунке 1.2. Три части схемы соответствуют трем выражениям (concurrent statements), описанным выше (порядок следования выражений в коде программы не имеет значения, поскольку данные выражения выполняются параллельно и могут следовать в любом порядке). Соединения между частями схемы однозначно определяются именами портов ввода/вывода и внутренних переменных.

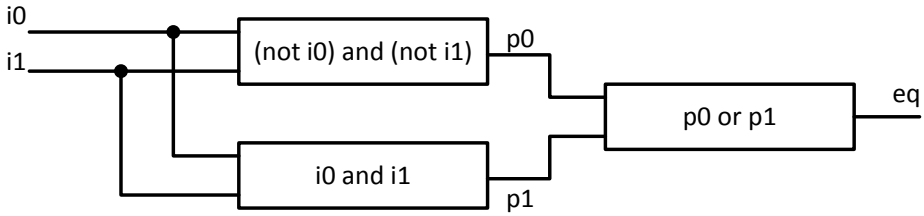

Рисунок 1.1 – Графическое представление реализации одноразрядной схемы сравнения

1.1.6 Двухразрядная схема сравнения

Увеличим разрядность схемы сравнения до двух. Пусть имеется входные 2-х разрядные сигналы *a* и *b* и выходной сигнал *aeqb*. Выходной сигнал *aeqb* принимает активное значение при одновременном равенстве и старших и младших разрядов *a* и *b*. Как и в случае одноразрядной схемы, описание реализуем в дизъюнктивной нормальной форме:

$$aeqb = \overline{a}_1 \cdot \overline{b}_1 \cdot \overline{a}_0 \cdot \overline{b}_0 + \overline{a}_1 \cdot \overline{b}_1 \cdot a_0 \cdot b_0 + a_1 \cdot b_1 \cdot \overline{a}_0 \cdot \overline{b}_0 + a_1 \cdot b_1 \cdot a_0 \cdot b_0,$$

где a_1, b_1 – старшие разряды сигналов *a* и *b* соответственно; a_0, b_0 – младшие разряды сигналов *a* и *b* соответственно;

Код реализации двухразрядной схемы сравнения приведен в листинге 1.2.

Листинг 2 – Реализация двухразрядной схемы сравнения на вентильном уровне

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity eq2 is
    Port ( a, b : in  STD_LOGIC_VECTOR (1 downto 0);
           aeqb : out  STD_LOGIC);
end eq2;

architecture sop_arch of eq2 is
      signal p0, p1, p2, p3 : STD_LOGIC;
begin
       -- сумма минтермов
       aeqb <= p0 or p1 or p2 or p3 ;
       -- определение минтермов
       p0 <= ((not a(1)) and (not b(1))) and ((not a(0)) and (not b(0))) ;
       p1 <= ((not a(1)) and (not b(1))) and ((a(0)) and (b(0))) ;
       p2 <= ((a(1)) and (b(1))) and ((not a(0)) and (not b(0))) ;
       p3 <= ((a(1)) and (b(1))) and ((a(0)) and (b(0))) ;
end sop_arch;
```

Входные порты *a* и *b* в данном случае объявляются как STD_LOGIC_VECTOR. Реализация структуры объекта аналогична реализации одноразрядной схемы – отличие только в количестве минтермов.

1.2 Структурное описание схем

Цифровые системы строятся зачастую на основе нескольких более простых унифицированных блоков. При этом на верхнем уровне иерархии описания цифровой системы структура данных унифицированных блоков не раскрывается – они представляются «черными ящиками» реализующими конкретные функции. Структура данных блоков раскрывается на более низком уровне иерархии описания системы. В свою очередь, каждый «черный ящик» может состоять из нескольких еще более простых унифицированных блоков. Структура этих унифицированных блоков раскрывается на еще более низком уровне иерархии описания системы и т.д. Такой подход, значительно упрощающий процесс проектирования и анализа сложных цифровых систем, получил название структурное или иерархическое описание схем. В языках описания аппаратуры HDL унифицированные блоки, которые используются в виде готовых объектов на верхнем уровне описания системы, получили название instance (образ объекта), а процедура размещения блока на верхнем уровне - instantiation.

В качестве альтернативы для реализации двухразрядной схемы сравнения (листинг 1.2) применим описанный выше структурный подход – в качестве унифицированного блока (instance) для построения двухразрядной схемы сравнения используется готовая одноразрядная схема (листинг 1.1). Структура двухразрядной схемы сравнения, реализованная указанным способом, представлена на рисунке 1.2. В приведенной структуре используется две одноразрядные схемы сравнения eq_bit0_unit и eq_bit1_unit, осуществляющие сравнение младших и старших бит входных сигналов соответственно и элемент *И*, на вход которого подаются сигналы с выходов указанных схем. Таким образом, выходной сигнал *aeqb* принимает активное значение при одновременном равенстве и старших и младших разрядов *a* и *b*.

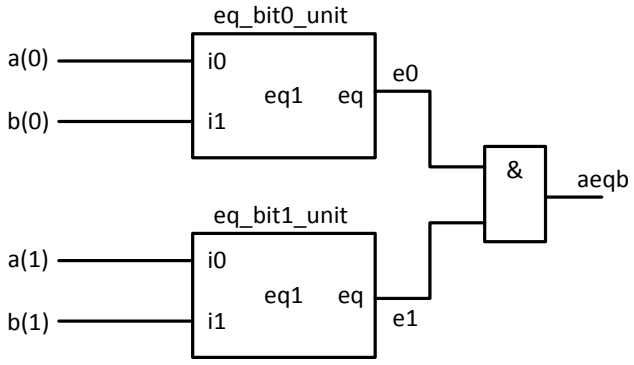

Рисунок 1Ошибка! Текст указанного стиля в документе отсутствует.**.2 – Реализация двухразрядной схемы сравнения на основе одноразрядных блоков**

Соответствующий код реализации двухразрядной схемы сравнения приведен в листинге 1.3 (Описание интерфейса (entity declaration) аналогично коду в листинге 1.2, поэтому в листинге 1.3 не приводится).

Листинг 3 – Описание структуры двухразрядной схемы сравнения

```
architecture struc_arch of eq2 is
      signal e0, e1 : STD_LOGIC;
begin
```

```
        -- вызов (instantiate) блока eq1 для сравнения младших битов
        eq_bit0_unit: entity work.eq1 (sop_arch)
              port map (i0 => a(0), i1 => b(0), eq => e0) ;
        -- вызов (instantiate) блока eq1 для сравнения старших битов
        eq_bit1_unit: entity work.eq1 (sop_arch)
              port map (i0 => a(1), i1 => b(1), eq => e1) ;
        aeqb <= e0 and e1 ;
        -- a и b равны, если равны одновременно и младшие и старшие биты
        aeqb <= e0 and e1 ;
end struc_arch;
```

Приведенный код (листинг 1.3) включает две процедуры вызова блока, реализующего одноразрядную схему сравнения (листинг 1.1). Рассмотрим подробно синтаксис процедуры вызова (instantiation statement) на примере вызова блока eq_bit0_unit:

```
eq_bit0_unit: entity work.eq1 (sop_arch)

      port map (i0 => a(0), i1 => b(0), eq => e0) ;
```

Процедура вызова начинается с объявления уникального имени вызываемого блока, в данном случае – **eq_bit0_unit**. Затем указывается название библиотеки (library name), в которой вызываемый блок находится – по умолчанию вызываемые блоки располагаются в рабочей библиотеке **work** создаваемого проекта. Идентификаторы **eq1** и **sop_arch** – имена интерфейса и архитектуры вызываемого блока соответственно (листинг 1.1). Вторая часть процедуры вызова заключается в привязке портов ввода вывода (port mapping) вызываемого блока к структуре проекта (рисунок 1.2). Таким образом, процедура вызова (instantiation statement) реализует схему, заключенную в «черный ящик», чья функция определяется в отдельном блоке (instance).

1.3 Упражнения

Схема «больше чем» сравнивает между собой 2 входа ***a*** и ***b*** и активизирует выход при условии, если ***a*** больше ***b***. Требуется разработать 4-х разрядную схему «больше чем» на вентильном уровне. Для этого:

1. Разработать таблицу истинности 2-х разрядной схемы «больше чем» и записать логику ее работы в совершенной дизъюнктивной нормальной форме. На основании полученного выражения, разработать VHDL-код, используя только логические операторы (вентили).
2. Разработать тестовый модуль (testbench) для моделирования работы 2-х разрядной схемы.
3. Используя 4 ползунковых переключателя для организации входа схемы и 1 светодиод для организации выхода, синтезировать схему и загрузить конфигурационный файл в отладочную плату. Протестировать схему.
4. Используя разработанную 2-х разрядную схему «больше чем», 2-х разрядную схему сравнения и минимальное количество дополнительной логики спроектировать 4-х разрядную схему «больше чем». Сначала нарисовать структурную схему, а затем, согласно схеме, разработать соответствующий VHDL-код.
5. Разработать тестовый модуль (testbench) для моделирования работы 4-х разрядной схемы.

6. Используя 8 ползунковых переключателя для организации входа схемы и 1 светодиод для организации выхода, синтезировать схему и загрузить конфигурационный файл в отладочную плату. Протестировать схему.

2 Проектирование комбинационных схем на регистровом уровне

Введение

В первой главе рассматривались простые схемы, описанные на вентильном уровне абстракции (gate level), с применением базовых логических вентилей, реализующих основные логические операции. В данной работе рассматривается проектирование схем средней степени интеграции, таких как сумматоры, дешифраторы, мультиплексоры и т.д. Описание подобных схем выполняется уже не на вентильном уровне абстракции (gate level), а на так называемом регистровом или функциональном уровне (RT-level). На данном уровне абстракции описывается поведение схемы (ее функция). В начале главы выполнен обзор операторов и конструкций, применяемых для описания схем на регистровом уровне.

2.1 Операторы и конструкции регистрового уровня описания схем

В дополнение к логическим операторам, рассмотренным в первой главе, для синтеза комбинационных схем применяются операторы сравнения (relational operators) и некоторые арифметические операторы (arithmetic operators). В данной главе рассматриваются данные операторы и разнообразные VHDL-конструкции на их основе. В таблицах 2.1 и 2.2 приведены основные операторы VHDL, применяемые для синтеза схем, и типы данных, с которыми данные операторы работают.

Таблица 2.2 – Операторы и соответствующие типы данных стандартной библиотеки VHDL-93 IEEE.STD_LOGIC_1164 package

<table>
<tr><th>оператор</th><th>описание</th><th>тип данных операндов</th><th>тип данных результата</th></tr>
<tr><td>a ** b</td><td>возведение в степень</td><td rowspan="5">Integer

тип integer используется для констант и границ массивов.
Не синтезируются!</td><td rowspan="5">integer</td></tr>
<tr><td>a * b</td><td>умножение</td></tr>
<tr><td>a / b</td><td>деление</td></tr>
<tr><td>a + b</td><td>суммирование</td></tr>
<tr><td>a - b</td><td>вычитание</td></tr>
<tr><td>a & b</td><td>конкатенация</td><td>1-D array, element</td><td>1-D array</td></tr>
<tr><td>a = b</td><td>равенство</td><td rowspan="2">любой</td><td rowspan="2">boolean</td></tr>
<tr><td>a /= b</td><td>не равенство</td></tr>
<tr><td>a < b</td><td>меньше чем</td><td rowspan="4">scalar или 1-D array</td><td rowspan="4">boolean</td></tr>
<tr><td>a <=b</td><td>меньше чем или равно</td></tr>
<tr><td>a > b</td><td>больше чем</td></tr>
<tr><td>a >=b</td><td>больше чем или равно</td></tr>
<tr><td>not a</td><td>отрицание</td><td rowspan="4">boolean, std_logic, std_logic_vector</td><td rowspan="4">такой же как у операндов</td></tr>
<tr><td>a and b</td><td>И</td></tr>
<tr><td>a or b</td><td>ИЛИ</td></tr>
<tr><td>a xor b</td><td>исключающее ИЛИ</td></tr>
</table>

Таблица 2.2 – Операторы и соответствующие типы данных библиотеки IEEE.NUMERIC_STD package

оператор	описание	тип данных операндов	тип данных результата
a * b	арифметические операции	unsigned, natural signed, integer	unsigned signed
a + b			
a - b			
a = b	операции сравнения	unsigned, natural signed, integer	boolean boolean
a /= b			
a < b			
a <=b			
a > b			
a >=b			

2.1.1 Операторы сравнения

В стандарте VHDL используется шесть операторов сравнения: **=** (равенство), **/=** (не равенство), **<** (меньше чем), **<=** (меньше чем или равно), **>** (больше чем), **>=** (больше чем или равно). Данные операторы выполняют операции сравнения между операндами одного типа и возвращают результат логического типа BOOLEAN. В данном учебном пособии тип данных BOOLEAN напрямую не используется, однако операции сравнения применяются в различных условных конструкциях VHLD, которые определяют структуру схемы при ее синтезировании. Подробно такие конструкции рассмотрены в разделе 2.2. Операторы сравнения синтезируются в виде компараторов.

2.1.2 Арифметические операторы

В стандарте VHDL арифметические операторы определены для типов данных INTEGER (целочисленный) и NATURAL (натуральные числа). Данные типы используются для констант и границ массивов, но не для синтеза.

Библиотека IEEE.NUMERIC_STD package

Для применение арифметических операторов при синтезе схем используется дополнительная библиотека IEEE.NUMERIC_STD package (таблица 2.2). Библиотека IEEE.NUMERIC_STD package добавляет два дополнительных типа данных: unsigned (беззнаковый) и signed (знаковый) и определяет операторы сравнения и арифметические операторы для этих типов данных. Типы данных unsigned и signed определены как массив элементов типа STD_LOGIC. Данный массив интерпретируется, как двоичное представление знакового или беззнакового целого числа. Для использования операторов из библиотеки IEEE.NUMERIC_STD package (таблица 2.2) ее нужно объявить вначале проекта:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;          -- объявление библиотеки numeric_std
```

Конвертация типов данных

Поскольку VHDL является языком со строгой типизацией, STD_LOGIC_VECTOR, unsigned и signed трактуются как различные типы данных, несмотря на то, что все они представляют собой массив элементов типа STD_LOGIC. Для конвертации данных из одного типа в другой используются функции конвертации (таблица 2.3). Обратите внимание, что данные типа STD_LOGIC_VECTOR представляют собой набор битов и не интерпретируются как число, поэтому данные типа STD_LOGIC_VECTOR не могут быть конвертированы в целочисленный тип INTEGER напрямую и наоборот.

Таблица 2.3 – Конвертация типов данных

Тип данных a	Конвертировать в …	Функция конвертации
unsigned, signed	std_logic_vector	std_logic_vector(a)
signed, std_logic_vector	unsigned	unsigned(a)
unsigned, std_logic_vector	signed	signed(a)
unsigned, signed	integer	to_integer(a)
natural	unsigned	to_unsigned(a, size)
integer	signed	to_signed(a, size)

Ниже приведены примеры правильной и неправильной конвертации типов данных. Пусть в проекте объявлены следующие данные:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
...
signal s1, s2, s3, s4, s5, s6 : STD_LOGIC_VECTOR(3 downto 0);
signal u1, u2, u3, u4, u5, u6 : UNSIGNED(3 downto 0);
...
```

Для начала рассмотрим следующие выражения:

```
u1 <= s1;       -- Ошибка типа (type mismatch)
u2 <= 5;-- Ошибка типа (type mismatch)
s2 <= u3;       -- Ошибка типа (type mismatch)
s3 <= 5;-- Ошибка типа (type mismatch)
```

Все приведенные выражения ошибочны. Необходимо данные, находящиеся в правой части выражений привести к типу данных, находящихся в левой части:

```
u1 <= unsigned(s1);                              -- Выражение верно
u2 <= to_unsigned(5, 4);                         -- Выражение верно
s2 <= std_logic_vector(u3);                      -- Выражение верно
s3 <= std_logic_vector(to_unsigned(5, 4));       -- Выражение верно
```

Обратите внимание, что для последнего выражения использовались две функции конвертации.

Теперь рассмотрим выражения, использующие арифметические операторы. Следующие выражения верны поскольку оператор «+» определен для типов данных unsigned и natural в библиотеке IEEE.NUMERIC_STD package:

```
u4 <= u1 + u2;          -- Выражение верно. Оба операнда типа unsigned
u4 <= u2 + 1;           -- Выражение верно. Операнды типа unsigned и natural
```

С другой стороны, следующие выражения являются неверными, поскольку для данных типа STD_LOGIC_VECTOR арифметические операции не определены:

```
s5 <= s1 + s2; -- Выражение не верно. Оператор «+» не определен для
```

```
                                используемых типов данных

s5 <= s2 + 1;  -- Выражение не верно. Оператор «+» не определен для

                                используемых типов данных
```

Для решения проблемы необходимо выполнить конвертацию операндов к типу unsigned (или signed), выполнить их сложение, а затем результат конвертировать обратно к типу STD_LOGIC_VECTOR:

```
S5 <= std_logic_vector(unsigned(s1) + unsigned(s2));          -- Выражение верно

S5 <= std_logic_vector(unsigned(s2) + 1);              -- Выражение верно
```

2.1.3 Другие синтезируемые VHDL конструкции

Оператор конкатенации

Оператор конкатенации «&» соединяет (склеивает) сегменты элементов в одну последовательность (массив). Следующий пример иллюстрирует использование конкатенации:

```
signal a1 : STD_LOGIC;
signal a4 : STD_LOGIC_VECTOR(3 downto 0);
signal b8, c8, d8 : STD_LOGIC_VECTOR(7 downto 0);
...
b8 <= a4 & a4;                          -- Соединяет два 4-х разрядных слова a4 в одно
                                           8-ми разрядное b8

c8 <= a1 & a1 & a4 & "00";               -- Соединяет два бита a1, 4-х разрядное слово
                                           a4 и 2-х разрядное слово «00» в одно 8-ми
                                           разрядное слово c8

d8 <= b8(3 downto 0) & c8(3 downto 0);-- Соединяет 4 младших разряда 8-ми разрядного
                                           слова b8 и 4 младших разряда 8-ми разрядного
                                           слова c8 в одно 8-ми разрядное слово d8
```

Оператор конкатенации «&» при синтезе схемы выполняется путем соединения входных и выходных сигналов, таким образом для его реализации требуются только «проводники».

Основное применение оператора конкатенации «&» - операции сдвига (shifting operations). В примере ниже оператор «&» применяется для реализации сдвига данных на фиксированное количество бит:

```
signal a : STD_LOGIC_VECTOR(7 downto 0);
signal rot, shl, sha : STD_LOGIC_VECTOR(7 downto 0);
signal b8, c8, d8 : STD_LOGIC_VECTOR(7 downto 0);
...
rot <= a(2 downto 0) & a(8 downto 3);       -- Сдвиг вправо на 3 бита

shl <= "000" & a(8 downto 3);               -- Сдвиг вправо на 3 бита с заполнением
                                            старших бит нулями (логический сдвиг)

sha <= a(8) & a(8) & a(8) & a(8 downto 3); -- Сдвиг вправо на 3 бита с расширением
                                            числа по знаку (арифметический сдвиг)
```

Z-состояние

Тип данных STD_LOGIC помимо значений ‘0’ и ’1’ содержит Z-состояние (высокоимпедансное состояние), характеризующее разомкнутую цепь. Z-состояние синтезируется исключительно с помощью трехстабильных буферов (tri-state buffer). Условное обозначение и таблица истинности трехстабильного буфера представлены на рисунке 2.1 и в таблице 2.4 соответственно.

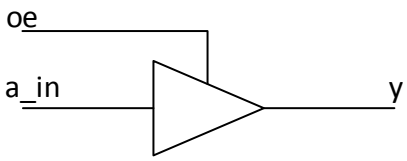

Рисунок 2.1 – Условное обозначение трехстабильного буфера

Таблица 2.4 – Таблица истинности трехстабильного буфера

oe	y
0	Z
1	a_in

Работа трехстабильного буфера контролируется входом разрешения oe (output enable) когда **oe** = ‘1’, входной сигнал передается на выход буфера, когда же **oe** = ‘0’ выходная цепь буфера разомкнута, т.е. буфер находится в Z-состоянии. Код, описывающий трехстабильный буфер выглядит следующим образом:

```
y  <= a_in when oe = '1' else 'Z';
```

Приведенное выше выражение называется «присваиванием по условия» и будет рассмотрено более подробно в разделе 2.2.1. Основное применение трехстабильного буфера – реализация двунаправленного порта ввода/вывода (bidirectional port) для более рационального использования контактов микросхем (I/O pins). Пример двунаправленного порта показан на рисунке 2.2.

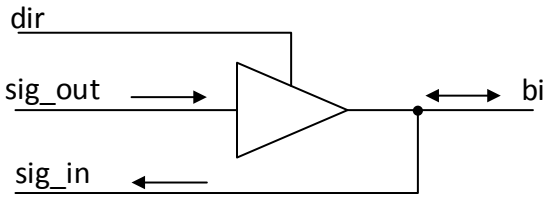

Рисунок 2.2 – Двунаправленный порт

Сигнал **dir** управляет направлением передачи данных. Когда **dir** = ‘0’ трехстабильный буфер находится в Z-состоянии и сигнал **sig_out** заблокирован – вывод **bi** используется как входной порт (**sig_in**). Когда **dir** = ‘1’ вывод **bi** используется как выходной порт и сигнал **sig_out** выдается во внешнюю цепь. VHDL-код, описывающий двунаправленный порт (рисунок 2.2) приведен ниже:

```
entity bi_demo is
    Port (bi : inout  STD_LOGIC;
          ...)
begin
      ...
      bi <= sig_out when dir='1' else 'Z';
      sig_in <= bi;
             ...
```

Обратите внимание что порт **bi** должен объявляться как inout, т.е. двунаправленный.

2.1.4 Выводы

VHLD – язык со строгой типизацией данных, поэтому при проектировании схем начинающими разработчиками часто можно видеть ошибку «Type mismatch». Ниже резюмируются основные правила, позволяющие избежать подобных ошибок. Поскольку данное учебное пособие нацелено прежде всего на синтезируемые конструкции VHLD, правила приводятся для синтезируемых типов данных и операторов:

- При объявлении входных и выходных портов (entity declaration) и внутренних сигналов (signals), не задействованных в арифметических операциях, используйте типы данных STD_LOGIC и STD_LOGIC_VECTOR.
- Используйте значение ‘Z’ только для синтезирования трехстабильных буферов.
- При объявлении внутренних сигналов (signals), участвующих в арифметических операциях, используйте библиотеку IEEE.NUMERIC_STD package и загружаемые с ней типы данных unsigned (беззнаковый) и signed (знаковый).
- Используйте функции конвертации (таблица 2.3) для преобразования данных типа STD_LOGIC_VECTOR в unsigned или signed и наоборот.
- Используйте встроенный тип данных integer и арифметические операции для задания констант и граничных значений массивов, но не для синтеза.
- Используйте результаты операторов сравнения, имеющих тип данных BOOLEAN в различных условных конструкциях VHLD (раздел 2.2), но не для синтеза.

2.2 Использование параллельных операторов присваивания в условных конструкциях VHLD

«Присваивание по условию» (conditional signal assignment) и «присваивание по выбору» (selected signal assignment) являются параллельными конструкциями (concurrent statements) в VHLD. Функционально данные конструкции напоминают операторы if и case в традиционных языках программирования, но в отличие от традиционных языков конструкции в VHLD выполняются параллельно, а не последовательно. Данные конструкции определяют структуру схемы при ее синтезировании.

2.2.1 Присваивание по условию (conditional signal assignment)

Синтаксис и синтезируемая структура

Синтаксис «присваивания по условию» приведен ниже:

```
signal_name <= value_expr_1 when boolean_expr_1 else
```

```
value_expr_2 when boolean-expr-2 else

. . .

value_expr_n;
```

Булевы выражения (**boolean_expr**) поочередно проверяются пока одно из них не оказывается истинным и соответствующее выражение (**value_expr**) присваивается сигналу **signal_name**. В случае если все булевы выражения ложны сигналу присваивается значение **value_expr_n**.

Структура «присваивание по условию» синтезируется в каскадную приоритетную схему на базе мультиплексоров 2 в 1. Условное обозначение и таблица истинности мультиплексора 2 в 1 показаны на рисунке 2.3 и таблице 2.5 соответственно.

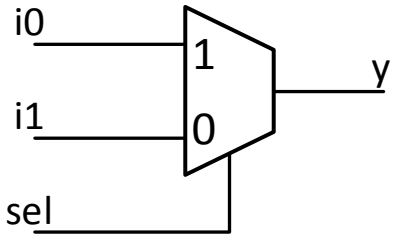

Рисунок 2.3 – Схема мультиплексора 2 в 1

Таблица 2.5 – Таблица истинности мультиплексора 2 в 1

sel	y
0 (false)	i0
1 (true)	i1

Рассмотрим пример синтеза следующего кода:

```
r <=  a + b + c when m = n else
      a - b     when m > n else
      c +1 ;
```

Каскадная приоритетная схема, в данном случае, реализуется с помощью соединения двух мультиплексоров 2 в 1 (рисунок 2.4).

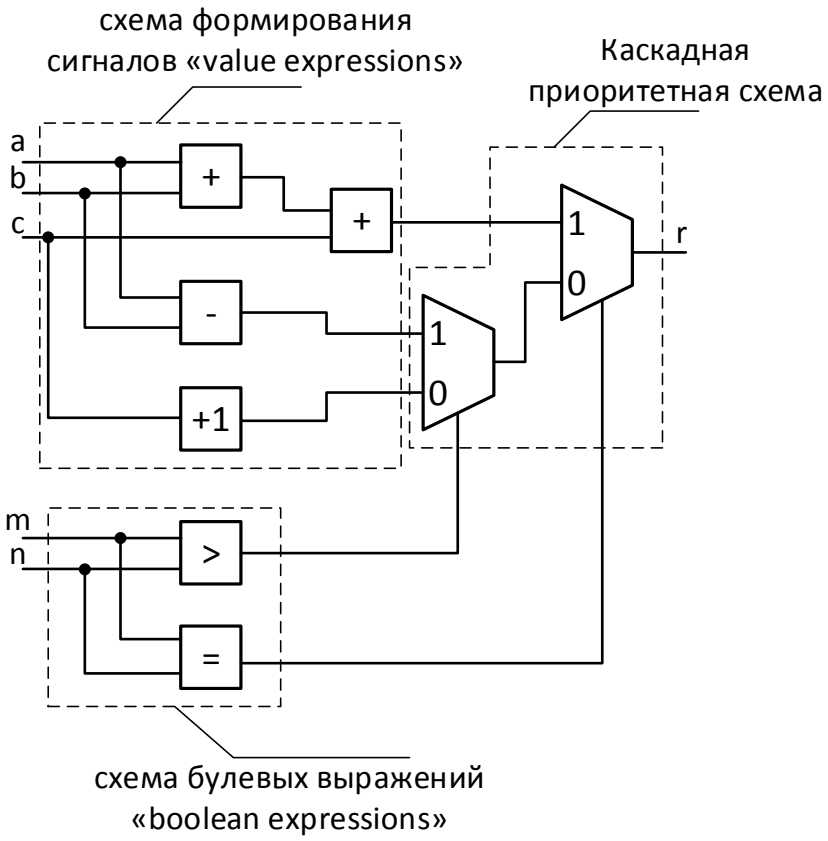

Рисунок 2.4 – Синтезированная схема, реализующая конструкцию «присваивание по условию»

Схема работает следующим образом: если истинно первое булево выражение (**m = n**), то сумма сигналов (**a + b + c**) передается на выход схемы **y**; если же истинно второе булево выражение (**m > n**), то на выход y передается разность сигналов (**a - b**); во всех остальных случаях (**m < n**) на выход схемы передается сигнал (**c + 1**).

Обратите внимание, что схема формирования сигналов (value expressions circuit) и схема булевых выражений (boolean expressions circuit) выполняются параллельно (рисунок 2.4). Выходные сигналы схемы булевых выражений используются как сигналы выбора в мультиплексорах, а выходные сигналы схемы формирования являются информационными сигналами, из которых выбирается необходимый сигнал и подается на выход схемы **r**. Количество ступеней (мультиплексоров) в каскадной приоритетной схеме (рисунок 2.4) соответствует количеству **when-else** условий в конструкции. Большое количество **when-else** условий приводит к синтезированию длинной цепи каскадов и, как следствие, к значительной временной задержке (propagation delay) работы схемы.

Ниже приведены два простых примера использования конструкции «присваивание по условию» (conditional signal assignment). Первым примером является приоритетный 4-х разрядный шифратор (priority encoder). Данный шифратор имеет четыре входных запроса (requests): **r(4)**, **r(3)**, **r(2)** и **r(1)**, которые объединены в 4-х разрядный вход **r**, причем старший бит **r(4)** имеет наивысший приоритет. Выходной сигнал **pcode** представляет собой двоичный код входного запроса с наивысшим приоритетом. Таблица истинности приоритетного шифратора приведена в таблице 2.6. Синтезируемый HDL код приведен в листинге 2.1.

Таблица 2.6 – Таблица истинности 4-х разрядного приоритетного шифратора

Input r	Output pcode
1×××	100
01××	011
001×	010
0001	001
0000	000

Листинг 2.1 – Синтез приоритетного шифратора с применение конструкции «присваивание по условию» (conditional signal assignment)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity prio_encoder is
    Port (
          r : in  STD_LOGIC_VECTOR (4 downto 1);
          pcode : out  STD_LOGIC_VECTOR (2 downto 0)
          );
end prio_encoder;

architecture cond_arch of prio_encoder is
begin
       pcode  <=  "100" when (r(4) = 1) else
                  "011" when (r(3) = 1) else
                  "010" when (r(2) = 1) else
                  "001" when (r(1) = 1) else
                  "000" ;
end cond_arch;
```

В приведенном коде (листинг 2.1) сначала проверяется состояние запроса r(4) и, в случае если он активен, на выход передается код “100”. Если запрос r(4) не активен проверяется состояние следующего по приоритету запроса, т.е. r(3) и т.д. пока не будут проверены остальные запросы.

Вторым примером использования конструкции «присваивание по условию» (conditional signal assignment) для синтеза схем является двоичный дешифратор. Двоичный дешифратор n-в-2^n преобразует двоичный n-разрядный код в унитарный 2^n-разрядный (код, в котором активным является только один бит). Таблица истинности двоичного дешифратора 2-в-4 приведена в
таблице 2.7. Схема дешифратора дополнительно имеет вход разрешения работы **en** (enable). Синтезируемый HDL код дешифратора приведен в листинге 2.2.

Таблица 2.7 – Таблица истинности двоичного дешифратора 2-в-4

Input			Output
en			y
a(1)		a(0)	
0	×	×	0000
1	0	0	0001
1	0	1	0010
1	1	0	0100

1	1	1	1000

Листинг 2.2 – Синтез двоичного дешифратора с применение конструкции «присваивание по условию» (conditional signal assignment)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decoder_2_4 is
    Port (
          a : in  STD_LOGIC_VECTOR (1 downto 0);
          en : in  STD_LOGIC;
          y : out  STD_LOGIC_VECTOR (3 downto 0)
          );
end prio_encoder;

architecture cond_arch of decoder_2_4 is
begin
       y  <=  "0000" when (en = '0') else
              "0001" when (a = "00") else
              "0010" when (a = "01") else
              "0100" when (a = "10") else
              "1000" ;     -- a = "11"
end cond_arch;
```

В приведенном коде (листинг 2.2) изначально проверяется состояние сигнала разрешения **en**, если сигнал активный, т.е. **en** = '1', то последовательно проверяются 4 входные комбинации сигнала **a**.

2.2.2 Присваивание по выбору (selected signal assignment)

Синтаксис и синтезируемая структура

Синтаксис «присваивания по выбору» приведен ниже:

```
with sel select
      sig <=    value_expr_1 when choice_1 ,

                value_expr_2 when choice_2 ,

                . . .

                value_expr_n when others;
```

Функционально «присваивание по выбору» напоминает оператор case в традиционных языках программирования. Данная конструкция присваивает необходимое значение (**value_expression**) выходному сигналу **sig** в зависимости от значения сигнала выбора **sel**. Каждый конкретный выбор (**choice_i**) представляет собой конкретное значение или набор

значений **sel**. Каждый выбор должен включать в себя любое значение **sel** не чаще одного раза, но при этом все вероятные значения **sel** должны присутствовать в структуре. Для отражения в структуре неиспользуемых вариантов используется служебное слово **others**, которое используется в обязательном порядке и включает в себя также не синтезируемые значения (‘X’, ’U’ и т.д.).

Структура «Присваивание по выбору» синтезируется в схему на базе мультиплексора. Рассмотрим пример синтеза следующего кода:

```
signal sel : STD_LOGIC_VECTOR (1 downto 0);

. . . .

r <=  a + b + c when “00” ,

      a - b     when “01” ,

      c +1      when others;
```

В данном случае сигнал **sel** может принимать 4 значения (“00”, ”01”, ”10”, “11”). Поэтому схема будет синтезирована в виде мультиплексора 4 в 1 с сигналом выбора **sel** (рисунок 2.5). Условное обозначение и таблица истинности мультиплексора 4 в 1 показаны на рисунке 2.6 и таблице 2.8 соответственно.

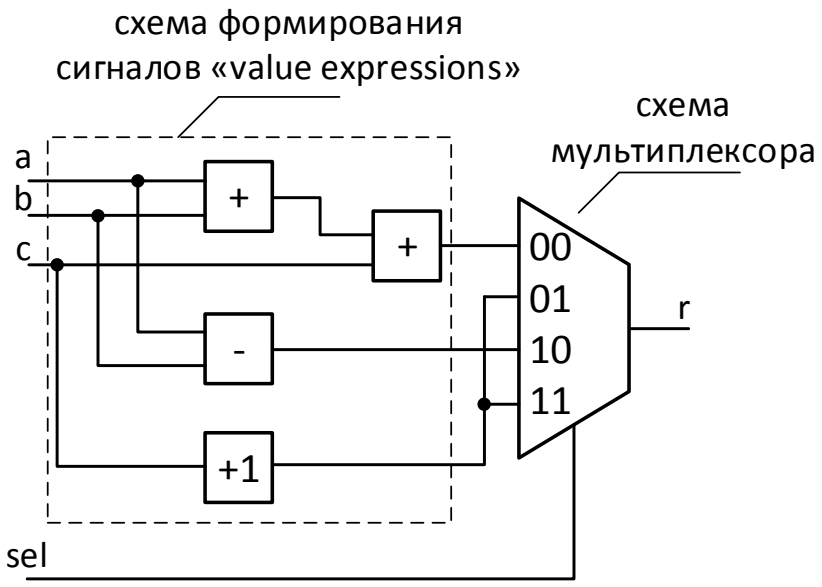

Рисунок 2.5 – Синтезированная схема, реализующая конструкцию «присваивание по выбору»

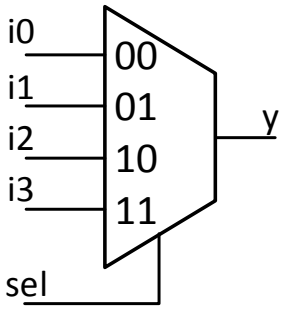

Рисунок 2.6 – Схема мультиплексора 4 в 1

Таблица 2.8 – Таблица истинности мультиплексора 4 в 1

sel	y
00	i0
01	i1
10	i2
11	i3

В результате схема (рисунок 2.5) работает следующим образом: сумма сигналов a+b+c передаётся на выход r при значении сигнала **sel** “00”; разность сигналов a-b передаётся на выход r при значении сигнала **sel** “10”; сигнал c+1 передаётся на выход схемы r при значении **sel** “01” и “11”.