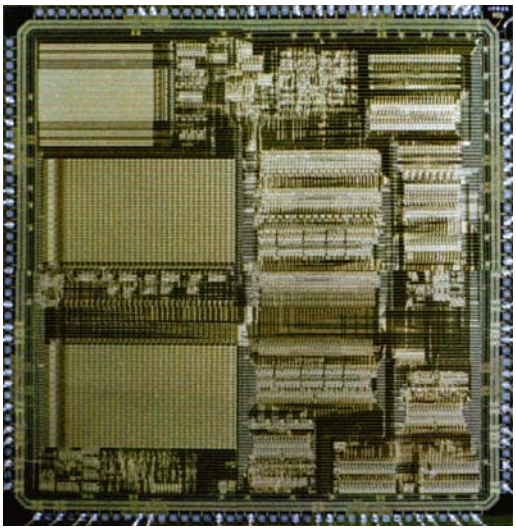
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

МОСКОВСКИЙ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

Ю.И. Бочаров, А.С. Гуменюк, А.Б. Симаков, П.А. Шевченко

ПРОЕКТИРОВАНИЕ БИС КЛАССА «СИСТЕМА НА КРИСТАЛЛЕ»

Рекомендовано УМО «Ядерные физика и технологии»
в качестве учебного пособия
для студентов высших учебных заведений



Москва 2008

УДК 621.3.049.771.16(075)
ББК 32.844.1я7
П79

Проектирование БИС класса «система на кристалле»: Учебное пособие / Ю.И. Бочаров, А.С. Гуменюк, А.Б. Симаков, П.А. Шевченко. – М.: МИФИ, 2008. 188 с.

Рассмотрены принципы построения и проектирования «систем на кристалле». Приведено пошаговое описание маршрута проектирования аналого-цифрового сложно-функционального блока по субмикронной КМОП технологии с использованием САПР компании Cadence.

Предназначено для студентов старших курсов, аспирантов и слушателей курсов повышения квалификации, обучающихся по направлениям микро- и наноэлектроника, информатика и вычислительная техника. Может быть полезно при изучении курсов «Системы автоматизированного проектирования в микроэлектронике», «Микросхемотехника», «Проектирование интегральных микросхем».

Учебное пособие подготовлено в рамках Инновационной образовательной программы

Рецензент д-р техн. наук, проф. И.И. Шагурин

ISBN 978 - 5 - 7262 - 1081 - 0

Предисловие

Появление больших интегральных схем (БИС) типа «система на кристалле» (СнК) было обусловлено тремя основными факторами. Первый из них – это развитие субмикронных технологий производства микросхем, позволившее достичь такой степени интеграции, которая позволила размещать на кристалле функционально очень сложные устройства. Второй фактор – огромные достижения в создании систем автоматизированного проектирования (САПР) микросхем высокой и сверхвысокой степени интеграции. Без наличия таких САПР проектирование «систем на кристалле» было бы принципиально невозможным, поскольку чрезмерно большая длительность периода от начала разработки до выхода изделий на рынок сделала бы не актуальным создание подобных БИС. К тому же весьма высокая степень вероятности появления ошибок при ручном или низкоуровневом автоматизированном проектировании не гарантирует надежной работы изготовленной БИС в системе. Очень важно то, что применение единых САПР позволило убрать барьер между разработчиками и потребителями СнК, и проектирование конечной аппаратуры ведется в их тесном взаимодействии. Третий фактор – появление реальной потребности в миниатюризации функционально сложных устройств. Появление этой потребности связано, прежде всего, с развитием рынка бытовой электроники и телекоммуникационных систем. Если специализированные интегральные микросхемы (ИМС) предыдущего поколения, реализуемые, например, на базовых матричных кристаллах, могли изготавливаться относительно малыми партиями для применения в специальной аппаратуре, то мелкосерийных специализированных «систем на кристалле» быть не может. Их разработка и изготовление комплекта фотошаблонов стоят слишком дорого. Именно непрерывно растущие потребности рынка бытовой, связной, а в последнее время и автомобильной электроники в конвергенции множества различных функций в минимальном объеме обусловили возможность крупносерийного производства таких сложных в разработке изделий как «системы на кристалле», сделав его экономически оправданным.

Помимо указанных факторов в развитии БИС типа СнК важную роль сыграло еще одно обстоятельство. Обычная практика исполь-

зования отработанных и проверенных технических решений в новых разработках, дополненная методологией применения САПР для высокоуровневого описания устройств и синтеза схем на основе таких описаний, привела к появлению концепции применения так называемых IP-блоков, которые в отечественной литературе и документации получили название сложнофункциональные (СФ) блоки [1]. Они играют роль основных строительных элементов или макрокомпонентов при создании СнК, значительно сокращая время и затраты на проектирование путем повторного использования готовых проверенных (верифицированных) технических решений. Этим, однако, концепция не исчерпывается. Не менее важным является то, что СФ блок (IP блок) это также предмет интеллектуальной собственности и рыночный продукт. Это отражено в его оригинальном названии: IP – Intellectual Property, интеллектуальная собственность – совокупность прав создателя и владельца блока. Поскольку использование результатов интеллектуальной деятельности, которые являются объектом исключительных прав, может осуществляться третьими лицами только с согласия правообладателя, это обеспечивает защиту. А поскольку СФ блок является рыночным продуктом, то его можно купить, продать или передать в той или иной форме. Возможность покупки готовых решений вместо того, чтобы тратить время и средства на их создание, с одной стороны, и возможность получить доход от плодов своей интеллектуальной деятельности, с другой стороны, оказала значительное стимулирующее влияние на развитие всей инфраструктуры, связанной с проектированием СФ блоков и «систем на кристалле».

Общепринятого определения такого термина как «система на кристалле» не существует, однако, основываясь на том, что было отмечено, можно определить его следующим образом. «Система на кристалле» – функционально законченная БИС, изготавливаемая по субмикронной технологии, разрабатываемая средствами САПР с широким использованием готовых СФ блоков и включающая процессорные элементы, схемы памяти и другие цифровые и аналого-цифровые узлы.

Анализу принципов построения, методов проектирования и тенденций развития «систем на кристалле» за рубежом и в нашей стране посвящен ряд отечественных публикаций [2 - 7], включая известную монографию [2]. Однако практически отсутствуют

учебные издания по этой тематике, поэтому авторы и решили написать данное пособие.

Книга логически и структурно делится на три раздела. В первом из них даются общие сведения о причинах, обусловивших появление БИС типа «система на кристалле», особенностях этого класса микросхем и общей стратегии их проектирования. Рассматривается методология использования СФ блоков в качестве готовых проверенных решений для повторного применения и в качестве продуктов интеллектуальной собственности. В связи с тем, что основным языком поведенческого описания цифровых СФ блоков является Verilog HDL, даются краткие сведения об основных конструкциях языка и примеры создания моделей устройств на языке Verilog. Рассматриваются особенности логического и физического синтеза цифровых схем на основе Verilog-описаний блоков, а также вопросы верификации проектов и тестирования БИС. Содержание первого раздела представлено, как краткий курс лекций, ориентированный на студентов старших курсов и аспирантов, имеющих необходимую подготовку в области схемотехники, проектирования ИМС и навыки использования современных САПР. Теоретический материал и практические примеры в основном не связаны с какой-либо конкретной платформой проектирования. Для выполнения практических работ по моделированию устройств с использованием языка Verilog подойдет любой доступный симулятор.

Во втором разделе кратко рассматриваются особенности проектирования аналого-цифровых блоков «систем на кристалле».

Третий раздел принципиально отличается от других. Он посвящен практическому освоению всех этапов маршрута проектирования аналого-цифровых блоков «систем на кристалле» – от получения технического задания до сдачи полного комплекта проектной документации на фабрику-изготовитель микросхемы. Фактически рассматривается процесс разработки 4-битного АЦП по КМОП технологии с проектными нормами 0,18 мкм. Теоретические сведения даются в ограниченном объеме. Поэтому для эффективного усвоения материала необходим достаточный уровень подготовки в области аналоговых и аналого-цифровых схем, измерительной техники, проектирования интегральных микросхем и знание операционной системы Linux. Вместе с тем сложные вопросы, часть кото-

рых не нашла отражения в учебной и в специальной литературе, рассмотрены подробно.

Важно отметить, что для практического выполнения полного цикла проектирования необходимы значительные ресурсы. Прежде всего, это наличие САПР заказных микросхем компании Cadence Design Systems (пакет IC версии 5.1 для ОС Linux), также наличие установленной проектной библиотеки одной из фабрик по производству КМОП БИС с проектными нормами 0,18 мкм, например XFAB или UMC.

Книга является обобщением опыта авторов по созданию БИС и сложнофункциональных блоков «систем на кристалле», проектирование которых было выполнено в МИФИ, ЗАО НТЦ «Модуль» и ИТМиВТ РАН им. С.А. Лебедева. Первый раздел пособия основан на курсе лекций, который читает один из авторов студентам 4-го курса, обучающимся на кафедре микро- и наноэлектроники МИФИ. Во второй и третий разделы вошло содержание методических пособий для студентов, выполняющих научно-исследовательскую работу и дипломное проектирование в центре проектирования специализированных БИС кафедр микро- и наноэлектроники и электроники МИФИ.

Книга адресована студентам старших курсов, обучающимся по направлениям электроника и микроэлектроника, информатика и вычислительная техника, а также по специальностям электроника и автоматика физических установок, микро- и наноэлектроника, системы автоматизированного проектирования. Ее можно рекомендовать аспирантам специальностей 05.27.01 – твердотельная электроника, радиоэлектронные компоненты, микро- и наноэлектроника, приборы на квантовых эффектах и 05.13.05 – элементы и устройства вычислительной техники и систем управления. Она также может быть полезной слушателям курсов повышения квалификации, переподготовки кадров и специалистам предприятий электронной отрасли и смежных областей.

Введение

Такие вещи, как мобильные телефоны и плееры, стали привычными, а, на самом деле, в них скрываются вычислительные системы высокой производительности. Активно развивается отрасль проводной и беспроводной связи. Требования к пропускной способности каналов связи непрерывно возрастают. Новые более эффективные алгоритмы требуют значительного повышения производительности вычислительных устройств. Таким образом, налицо тенденция непрерывно возрастающей потребности в повышении производительности вычислительных систем. При этом энергопотребление должно оставаться на прежнем уровне или даже уменьшаться, чтобы увеличить время непрерывной автономной работы носимых устройств.

Коммерческие продукты обеспечивают большие объемы выпуска изделий, но вместе с тем требования к элементной базе могут значительно различаться для разных серийных продуктов. Необходимо максимально сокращать время разработки, что влечет за собой необходимость возможно большего использования предыдущих наработок. Отсюда возникает тенденция к появлению аппаратных платформ, обеспечивающих быструю разработку системы из готовых блоков, как из конструктора. При этом нет необходимости создавать систему заново, с пустого места.

Очевидная тенденция развития современной электроники – постоянное уменьшение размеров устройств, что требует также постоянной минимизации элементной базы. Это, в свою очередь, влечет за собой необходимость размещения возможно большого числа компонентов устройства в одном корпусе, а в предельном случае – всей системы в одном корпусе, на одном кристалле. Такие системы обычно и называют «системами на кристалле».

В настоящее время большинство потребительских электронных устройств строится на однокристальном варианте системы, за исключением высокочастотных блоков обработки радиосигналов, где часто применяют технологии на основе арсенида галлия, а также периферийных устройств с более высокими, чем у основной микросхемы, уровнями напряжения.

И только самые новые изделия создаются на нескольких микросхемах по той причине, что на одном чипе все блоки не помещают-

ся, но проблема решается с появлением новых технологий с меньшими технологическими нормами.

Отдельно нужно выделить класс систем в корпусе (System in Package – SiP), в основе которых лежит технология помещения в один корпус нескольких кристаллов, возможно, разной природы, например кремниевый кристалл с цифровым процессором сигналов и кристалл арсенида галлия с высокочастотным радиотрактом. Что имеет смысл собирать в один корпус? То, что невозможно собрать на одном кристалле, потому что разные технологии – высокочастотные аналоговые схемы, выполненные на арсениде галлия, энергонезависимая память, батарейка и т. д.

«Системы на кристалле» характеризуются платформой, на основе которой они построены. Известных платформ не так уж и много. Объясняется это тем, что каждая платформа определяет интерфейс связи отдельных блоков внутри СнК. Интерфейс этот должен быть задан строго и однозначно, так как разработкой отдельных блоков могут заниматься разные компании. Кроме того, платформа определяется основным системным процессором, который осуществляет управление устройством.

Классифицировать «системы на кристалле» можно по степени универсализации. Цифровые сигнальные процессоры и процессоры общего назначения можно считать самыми универсальными СнК. Следующую группу составляют устройства, специализированные для применения на некотором, ограниченном классе задач, например СнК для обработки радиолокационных сигналов или системы на кристалле для обработки аудиосигналов. Наиболее специализированными являются СнК, разработанные для выполнения определенной задачи и предназначенные для применения в определенном устройстве или в нескольких типах устройств, например в мобильных телефонах одной серии.

Типичная система на кристалле состоит из основного ведущего процессора, который обеспечивает управление всей системой, а также может выполнять определенные вычисления; блоков цифровой обработки сигналов (сигнальных процессоров и блоков «жесткой» логики, аппаратно выполняющих алгоритм); блоков памяти; аналого-цифровых узлов и многочисленных интерфейсных блоков.

В качестве примера на рис. В.1 приведена структурная схема и показан состав типичной цифровой «системы на кристалле». Это

специализированная БИС навигационного приемника Глонасс/GPS, разработанная предприятием ФГУП НИИМА «Прогресс» [6]. Устройство предназначено для корреляционной обработки шумоподобных навигационных сигналов, формирования меток времени и решения навигационной задачи.

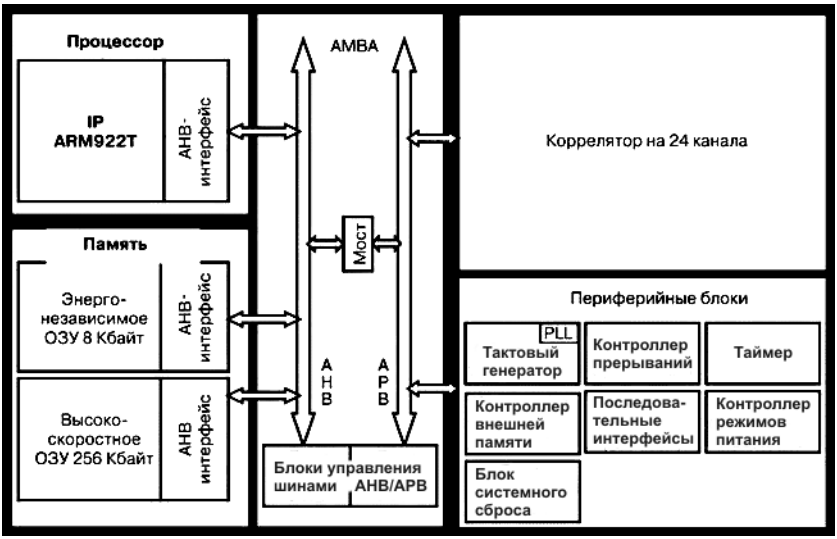


Рис. В.1. Состав типичной цифровой «системы на кристалле»

Основой устройства является специально разработанный блок 24-канального коррелятора для декодирования сигналов. В качестве процессора, решающего навигационную задачу и осуществляющего управление системой, использован СФ блок RISC-процессора ARM922T, приобретенный у компании ARM. СФ блок включает 32-разрядное ядро процессора и шину AMBA. Другие блоки СнК разработаны на основе библиотеки стандартных ячеек компании UMC. Микросхема изготовлена на фабрике UMC по технологии с проектными нормами 0,18 мкм. Помимо вычислительных и управляющих блоков она содержит кэш-память объемом 16 Кбайт, статическое ОЗУ емкостью 256 Кбайт, энергонезависимые часы, ОЗУ, последовательные интерфейсы, контроллер внешней памяти и бло-

ки тестирования. Степень интеграции – около восьми миллионов транзисторов.

Структура типичной аналого-цифровой СнК показана на рис. В.2. Это декодер цифрового телевизионного сигнала высокой четкости, разработанный компанией ЗАО НТЦ «Модуль» [7].

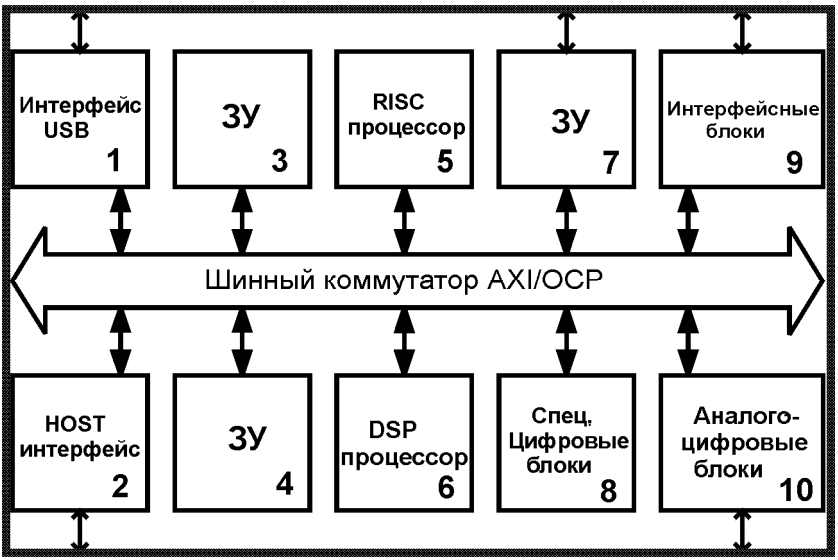


Рис. В.2. Состав аналого-цифровой СнК

Пояснения к структурной схеме «системы на кристалле» даются со ссылками на номера блоков, показанных на рис. В.2.

Система построена преимущественно на базе СФ блоков, поставляемых компанией Fujitsu. Это основной RISC-процессор *5* с ядром ARM1176; блоки, составляющие инфраструктуру шин данных формата AMBA; аналого-цифровые блоки *10* – ЦАП и тактовый генератор с фазовой автоподстройкой частоты; контроллеры интерфейсов USB, Ethernet, HDMI, UART *1, 2, 9* и др. Специализированные блоки *6, 8,* предназначенные для декодирования видеосигнала – собственная разработка ЗАО НТЦ «Модуль». Микросхема изготавливается по технологии КМОП фирмы Fujitsu с проектными нормами 90 нм.

Главное в методологии проектирования понимание того, что СнК – система и разработка начинается с системного уровня. В первую очередь, необходимо определить класс задач или устройство, для которого разрабатывается СнК.

Как правило, имеется прототип. Он существует в виде многокомпонентной системы на одной или нескольких печатных платах. Цель создания прототипа состоит в предварительной оценке структуры, опробовании функционирования устройства, согласовании внешних интерфейсов и отладки программного обеспечения системы. Задача разработчика СнК перенести эту систему внутрь кристалла, как минимум, сохранив, а лучше увеличив ее производительность.

Маршрут проектирования СнК такой же, как для СБИС, изготавливаемых по субмикронным технологиям. Он включает следующие этапы:

- создание модели системного уровня и тестов функционального контроля микросхемы;
- разработка моделей блоков и всего устройства на уровне регистровых пересылок (RTL);
- логический и физический синтез отдельных блоков и микросхемы, в целом;
- разработка системы аппаратного тестирования микросхемы и тестов аппаратного контроля;
- проектирование топологии микросхемы;
- изготовление микросхемы;
- тестирование и испытания готовых микросхем.

Определяющим фактором в разработке программного обеспечения является тип ведущего процессора. Как правило, в качестве основного процессора выбирается устройство с широко известной и используемой системой команд. В этом случае основное программное обеспечение мало зависит от структуры системы и может легко переноситься с одной СнК на другую, в которой применен тот же ведущий процессор. Обеспечивается возможность использования программного обеспечения сторонней разработки, что значительно сокращает и облегчает работу. Для всех наиболее часто используемых платформ существует множество библиотек при-

кладных программ, как коммерческой разработки, так и находящихся в открытом доступе.

Каждая СнК имеет свой уникальный набор дополнительных СФ блоков, начиная с устройств цифровой обработки сигналов и заканчивая различными периферийными и интерфейсными устройствами. Работа с функциями и устройствами, уникальными для данной системы, осуществляется специализированным программным обеспечением (ПО), которое может создаваться для каждого проекта в отдельности. Зачастую именно специализированное ПО отличает одни СнК от других. Как правило, работой системы управляет операционная система. Соответственно организация управления периферийными устройствами сводится к написанию драйверов устройств, которые для общеупотребительных интерфейсов вполне стандартны.

Разработка СнК неотделима от разработки устройств на основе этих микросхем. СнК разрабатывается для вполне определенного приложения и должна функционировать во вполне определенном устройстве или в семействе устройств. В них помимо СнК присутствуют и другие элементы, взаимодействие и интерфейс с которыми должны быть обеспечены в полном объеме, т.е. разработка устройства на основе системы на кристалле должна вестись одновременно с разработкой самой микросхемы. Только так возможно добиться оптимизации системы, а соответственно, ее удешевления, что очень важно для коммерческих устройств.

1. Методология проектирования цифровых блоков СнК

1.1. Технологии разработки и производства СнК

Из всего многообразия современных технологий производства интегральных схем наибольшее развитие получила технология КМОП. На текущий момент только она позволяет разместить на одном кристалле десятки миллионов логических вентилей при технологических нормах проектирования 90 нм. В дальнейшем, упоминая технологию создания СнК, будем подразумевать КМОП технологию.

В зависимости от маршрута проектирования можно выделить следующие виды технологии проектирования:

- полностью заказная технология, где топология интегральной схемы целиком определяется разработчиком, включая топологии базовых элементов, транзисторов и др.;
- полузаказная технология, где разработчику предоставляются библиотеки таких базовых элементов, как транзисторы, на основе которых можно создавать собственные проекты;
- технология на основе библиотек логических элементов (стандартных ячеек);
- технология проектирования на основе базовых матричных кристаллов различного уровня детализации и с использованием программируемых логических интегральных схем (ПЛИС).

Заказные и полузаказные технологии разработки являются наиболее трудоемкими. Время проектирования логических устройств может превышать время, затрачиваемое при использовании других маршрутов проектирования, в десятки раз. Такие разработки могут позволить себе только очень крупные электронные фирмы и только для создания наиболее совершенных изделий. Во всех прочих случаях заказные и полузаказные технологии используются для создания небольших устройств, которые применяются в качестве библиотечных элементов в других, более простых, маршрутах проектирования. На основе заказных и полузаказных технологий создаются аналоговые узлы и блоки памяти. Сложность работы с заказными технологиями – трудности с верификацией результатов. Моделирование устройств может быть осуществлено только на уровне

транзисторов или даже топологических примитивов, что означает необходимость решения систем дифференциальных уравнений. Соответственно, подобное моделирование, хотя и может дать весьма точные данные о работе устройства, осуществляется очень медленно и физически невозможно промоделировать устройство очень большого объема. Системы автоматизированного проектирования заказных и полузаказных схем – одни из самых дорогих.

Гораздо проще и удобнее в использовании технология проектирования на основе библиотек логических элементов. Библиотечными элементами могут выступать как простейшие блоки, выполняющие элементарные логические функции, так и более сложные многовходовые логические элементы и триггеры. Библиотечные элементы различаются как по виду выполняемых функций, так и по нагрузочной способности выходов элемента. Кроме простых логических элементов, при проектировании могут использоваться более сложные конструкции и макроблоки: аналого-цифровые устройства, блоки памяти, другие блоки, разработанные по заказной технологии проектирования. Отдельными видами библиотечных элементов являются внешние буферы ввода-вывода. Характеристики библиотечных элементов определяются при их проектировании и поставляются разработчику в виде библиотек для использования в различных системах автоматизированного проектирования.

Стандартные библиотечные элементы имеют одинаковую высоту и располагаются на кристалле горизонтальными рядами. Расстояние между рядами, как правило, фиксировано. Расстояние между ячейками в ряду кратно шагу трассировки для данной технологии. Таким образом, размещая элементы на некотором расстоянии друг от друга, можно обеспечить прохождение необходимого числа трасс металлизации. Такие макроблоки, как блоки памяти, имеют размеры большие, чем высота ряда ячеек, и располагаются на кристалле независимым образом.

Кроме технологии проектирования на основе стандартных ячеек, существует ряд технологий, обеспечивающих ускоренный маршрут проектирования за счет предварительной разработки и изготовления структурных элементов системы. Это технологии на основе базовых матричных кристаллов или «моря вентилей» и технологии проектирования на основе программируемых логических интегральных схем ПЛИС.

Элементы библиотеки стандартных ячеек строятся на одних и тех же транзисторных структурах. Если эти структуры изготовить на кристалле заранее, то функцию самого элемента можно определить в процессе формирования слоев металлизации. Металлизация определяет и порядок соединения элементов между собой. Таким образом, для проектирования системы необходимо разрабатывать только фотошаблоны, отвечающие за слои металла, что значительно удешевляет процесс разработки и производства. Создать систему по подобной технологии можно в наиболее короткие сроки. Также можно изготавливать микросхемы меньшими партиями.

Недостаток такой системы проектирования и разработки состоит в том, в чем заключено и ее основное преимущество – базовые транзисторные структуры изготовлены заранее. Это означает, что невозможно использовать другие базовые структуры, если возникнет такая необходимость, например, для создания блоков памяти или аналого-цифровых блоков. На основе цифровых базовых структур эти блоки реализуются крайне неэффективно. Возможное решение – включение блоков в состав базового кристалла на этапе его создания. Однако это означает, что базовый кристалл изготавливается для определенной системы и большая часть преимуществ технологии быстрого проектирования теряется. Единственным положительным фактором остается возможность относительно простого внесения исправлений в систему при условии, что базовый кристалл остается прежним. Для этого необходимо внести исправления только в слои металлизации.

1.2. Маршрут проектирования

Маршрут проектирования СнК включает в себя следующие стадии: постановка задачи и составление технического задания, создание алгоритмической и функциональной моделей системы, разработка модели на уровне регистровых пересылок, разработка принципиальной электрической схемы (в заданном технологическом базисе), проектирование топологии, изготовление и тестирование СнК. На рис. 1 представлены основные этапы маршрута создания СнК с указанием взаимных связей между ними, а также информационные ресурсы и средства разработки (моделирования, синтеза, верификации), используемые на различных этапах.

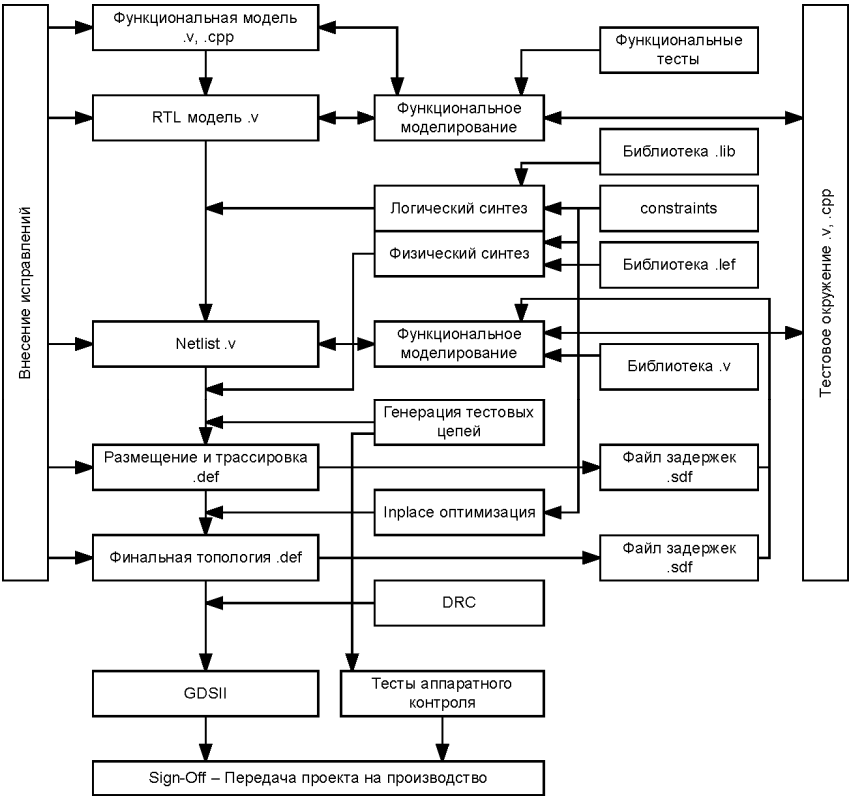


Рис. 1. Маршрут проектирования СнК

Разработка системы на кристалле – это, прежде всего, построение системы на основе набора готовых элементов – сложнофункциональных блоков. Большей частью используются готовые, неоднократно испытанные в различных проектах решения. Разрабатываются лишь специализированные блоки. Рассмотрим типовой маршрут проектирования СнК подробнее.

На этапе постановки задачи и составления технического задания определяется функциональность микросхемы.

Алгоритмическая и функциональная модели определяют общее функционирование устройства и используются для оценки необхо-

димых параметров будущей системы. Модели могут создаваться на программном языке, например, C++ или в такой специализированной среде моделирования, как Matlab Simulink. Алгоритмические и функциональные модели покрывают, как правило, не всю функциональность устройства, а только его основные функции.

Основным этапом разработки является создание модели системы на уровне регистровых пересылок – так называемой RTL модели. На данном уровне описания определяется иерархия системы. Описание включает все имеющиеся в системе регистры и комбинационную логику между ними в виде логических выражений на одном из языков описания аппаратуры. Широко используются три из них: SystemC, Verilog и VHDL. Каждый из языков имеет свои преимущества и свою нишу по использованию. Современные системы моделирования поддерживают работу, как правило, со всеми тремя языками сразу в режиме смешанного моделирования. Разработка RTL модели системы включает в себя сборку иерархической модели на основе готовых моделей блоков.

Важно отметить, что этот этап разработки является технологически независимым, т.е. RTL код не зависит от одной определенной технологии и может быть перенесен на любой заданный технологический базис в процессе синтеза. С другой стороны, RTL код уже содержит информацию обо всех последовательностных элементах системы и комбинационной логике между ними. Так как частота работы устройства напрямую зависит от объема логики между регистрами, RTL код определяет некоторый диапазон частот, который может быть достигнут при переносе системы на заданную технологию, т.е. основная оптимизация параметров устройства осуществляется еще на этапе написания RTL кода. Моделирование RTL кода осуществляется, как правило, с точностью до процессорного такта синхронных устройств.

Одновременно с написанием RTL кода формируется верификационное окружение модели. Верификационное окружение предназначено для проверки правильности работы модели. В зависимости от сложности системы сложность верификационного окружения может значительно изменяться – от возможности подачи простых наборов входных сигналов до построения параллельной функциональной модели устройства, служащей эталоном в процессе верификации.

В процессе написания RTL кода в систему добавляются такие блоки, обеспечивающие дальнейшее тестирование микросхем, как контроллер тестового порта, устройства автоматического тестирования памяти и других функциональных блоков, также добавляются необходимые тестовые выводы.

Работу с RTL кодом поддерживают большинство пакетов САПР, в том числе Cadence IUS, Synopsys VCS/VSS, Mentor ModelSim, Aldec ActiveHDL и многие др. Особенности работы с определенной САПР можно узнать из соответствующей пользовательской документации.

На следующем этапе осуществляется перенос RTL кода СнК на заданную технологию. Выполняется он, как правило, путем логического или физического синтеза в САПР. В результате логического синтеза абстрактные последовательностные и комбинационные элементы, описанные в RTL коде, заменяются на элементы из технологической библиотеки. В ходе оптимизации подбираются библиотечные элементы, параметры которых наиболее оптимально соответствуют заданным характеристикам системы. Сами характеристики задаются в специальных файлах, так называемых файлах ограничений (constraints).

В зависимости от выбранной целевой технологии емкости связей между библиотечными элементами могут вносить значительный вклад во временные характеристики системы. Начиная с технологии 0,25 мкм и далее, для правильного определения временных параметров системы нужно знать длины связей между элементами. Для этого элементы необходимо разместить на кристалле. Синтез электрической схемы, при котором производится размещение элементов на кристалле, называется физическим. Физический синтез позволяет получить схему с лучшими характеристиками, но требует значительно больше аппаратных ресурсов.

Синтез схемы – процесс итеративный. Он может выполняться как для отдельных блоков системы, так и для системы в целом неоднократно – до тех пор, пока не будет достигнут приемлемый результат. На основе полученных результатов синтеза может производиться изменение логики устройств и RTL кода.

К синтезированной схеме добавляются блоки аппаратного тестирования, регистры объединяются в сканирующие цепи, подключается контроллер тестового порта, а также схемы автоматического

тестирования отдельных устройств, если они не были подключены ранее. Схемы автоматического тестирования подключаются и организуются, как правило, в автоматическом режиме, при помощи специальной САПР. Одновременно с подключением тестовых цепей генерируются тестовые последовательности сигналов, позволяющие при помощи этих тестовых цепей проверять правильность функционирования изготовленной микросхемы.

Следует отметить принципиальную разницу в назначении тестов. Функциональные тесты пишутся разработчиками микросхемы и проверяют, что микросхема имеет заданную функциональность, т.е. выполняет все требуемые задачи во всех заданных режимах.

Тесты функционального контроля, как правило, генерируются в автоматическом режиме и предназначены для проверки правильности изготовления микросхемы, т.е. что готовая микросхема логически эквивалентна микросхеме, переданной на производство в виде проекта.

На основе синтезированной схемы производится разработка топологии СнК. Кроме непосредственно размещения элементов и трассировки межсоединений, которые могут выполняться также и в процессе физического синтеза, производится множество других подготовительных операций. В частности, создается детальный план кристалла, разрабатывается порядок размещения внешних буферов ввода-вывода по периферии чипа (цоколевка), производится анализ энергопотребления схемы и проектируется сетка шин земли и питания (power plan), разрабатываются и размещаются цепи синхронизации – «дерево» тактовых сигналов (clock tree).

Созданный топологический проект подвергается тщательному анализу. Исследуются временные параметры микросхемы, которые по результатам топологического проектирования могут быть оценены со значительно большей точностью.

Исследуется большой набор различных параметров микросхемы, включая уровни потребляемой мощности блоков, максимальное число нагрузок элементов, допустимые значения емкостей соединений, времен фронтов сигналов, наличие потенциальных наводок сигналов (crosstalk) и т.д. Объем исследуемых параметров и диапазон необходимых значений определяется заданной целевой технологией. Чем новее и совершеннее технология, чем дальше в

субмикронную область приходится уходить, тем большее число факторов приходится учитывать.

Допустимые значения параметров задаются для определенной технологии фабрикой-изготовителем и обязательно контролируются на входном контроле при передаче проекта на изготовление. Только так можно обеспечить возможность изготовления микросхемы с теми характеристиками, которые от нее ожидаются.

Кроме того, обязательно производится проверка проекта на соответствие набору формальных правил (Design Rule Check, DRC). Этот набор правил также диктуется кремниевой фабрикой и определяет формальные требования к проекту, начиная от требований к написанию имен сигналов, построению иерархии системы, до правил размещения элементов на кристалле.

Требования DRC необходимы для обеспечения корректной работы программного обеспечения на кремниевой фабрике в процессе изготовления фотошаблонов и проведения испытаний микросхем.

Как правило, параметры микросхемы начинают соответствовать всем необходимым значениям далеко не сразу. Разработка – долгий итеративный процесс, в котором в зависимости от сложности микросхемы и величины выявленных расхождений приходится возвращаться к предыдущим стадиям проектирования.

Так, если несоответствие временных параметров – небольшое, можно провести коррекцию критичных цепей микросхемы, не изменяя топологии, путем изменения лишь нескольких отдельных элементов. В случае более значительных несоответствий может потребоваться новый синтез схемы или даже внесение изменений в RTL код.

Конечно же, чем дальше зашел процесс разработки, тем дольше и дороже вносить в систему изменения. Для того чтобы упростить процесс, в современных САПР появились средства быстрого прототипирования системы, т.е. САПР позволяет в минимальные сроки создать первоначальный вариант топологии системы, чтобы анализ параметров и необходимая коррекция производились на возможно более раннем этапе разработки. Второй важно тенденцией является максимальная автоматизация процесса разработки путем использования различных командных файлов. Такой подход позволяет избежать двойной ручной работы. То, что однократно

было сделано, второй раз может быть быстро переделано в автоматическом режиме.

Финальной стадией разработки микросхемы является передача проекта на производство. Осуществляется изготовление фотошаблонов, производство микросхем. Затем микросхемы тестируются с использованием тестов аппаратного контроля. Годные чипы корпусируются и проходят те или иные процедуры испытаний в зависимости от предъявляемых к ним требований.

Маршрут разработки микросхем разбивается на несколько стадий: алгоритмическое и функциональное проектирование; логическое проектирование; физическое проектирование; разработка тестов; изготовление микросхем и испытания микросхем. В зависимости от того, кем выполняются отдельные стадии разработки, возможны различные схемы разделения труда.

Например, разработка может доводиться до стадии логического проектирования, а физическое проектирование может выполняться дизайн-центром, который обслуживает производителя микросхем. Такая схема имеет широкое распространение и называется fables, т.е. разработка без производства конечного продукта на фабрике.

Нередко сторонними разработчиками выполняется разработка тестов и тестового окружения. Такой подход позволяет значительно повысить качество и эффективность тестирования и обнаруживать ошибки в системе, пропущенные разработчиками логических блоков.

Разработка СнК разделяется на два этапа: разработка компонентов системы (СФ блоков) и интеграция этих блоков в систему на кристалле.

Лидерами в создании САПР для проектирования микросхем уровня СнК в настоящее время являются фирмы Cadence и Synopsys. Они предлагают набор программного обеспечения (ПО), покрывающий весь маршрут проектирования от алгоритмических моделей до создания технологических файлов в формате GDSII, передаваемых на производство. Имеется также ряд компаний, разрабатывающих программы для отдельных этапов разработки проекта.

Маршрут проектирования для каждой компании разработчика СБИС – сложный многоступенчатый процесс, предполагающий использование большого числа компонентов различных фирм, а

также продуктов собственной разработки, создание библиотек скриптов для выполнения типичных действий и т.д. Он является уникальным ноу-хау компании, позволяющим ей разрабатывать конкурентноспособную продукцию.

1.3. Особенности проектирования с использованием сложнофункциональных блоков

Описанный маршрут проектирования незначительно отличается от маршрута проектирования любых других микросхем. Отличие, которое является принципиальным, – это использование большого числа готовых СФ блоков (IP блоков).

Ключевым фактором в данном случае является именно IP (Intellectual Property) – интеллектуальная собственность. Каждый блок является объектом интеллектуальной собственности, имеет свою, порой весьма высокую, стоимость. Использование IP блоков предусматривает набор механизмов, обеспечивающих возможность защиты блоков от несанкционированного использования и, что не менее важно, от несанкционированного изменения. Блоки могут подключаться в систему на различных уровнях проектирования и могут быть представлены в следующих формах.

Открытый RTL код (soft macro). В этом случае блок полностью доступен для работы, в том числе для внесения изменений. С другой стороны, ответственность за правильность функционирования блока после синтеза и физического проектирования полностью ложится на разработчиков СнК.

Закрытый (зашифрованный) RTL код. В этом случае внесения изменений в логический код невозможны. Как правило, зашифрованный логический код доступен для логического и физического синтеза лишь в определенных САПР.

Электрическую схему или список цепей часто называют нетлистом – термином, близким по звучанию и по смыслу соответствующему англоязычному термину (netlist). Блок заранее синтезирован для заданной технологии. Внесение изменений в код в данном случае практически невозможно. Такой подход позволяет разработчикам использовать СФ блок, не углубляясь в его внутреннюю структуру. Характеристики синтезированной схемы могут быть лучше. С другой стороны, такой вариант поставки возможен

только, если поставщих СФ блока имеет доступ к целевой технологии. Кроме того, логическое моделирование электрической схемы может быть слишком медленным. Как правило, вместе со схемой в таком случае поставляется еще и быстрая симуляционная модель блока.

Топология блока (hard macro). В этом варианте топология создается для заданной целевой технологии. Разработчикам СнК остается только включить блок в нужное место системы. Никакие его модификации невозможны, в том числе и изменение топологических параметров. Такой метод поставки характерен для fables маршрута проектирования микросхем. В этом случае топология блока вовсе не поставляется разработчикам СнК, а остается в ведении дизайн-центра, осуществляющего физическое проектирование. Для выполнения логического проектирования и моделирования поставляется симуляционная модель и набор библиотек с параметрами блока. Для разработчика СнК блок выглядит как любой элемент из библиотеки стандартных ячеек.

Обобщая сказанное, можно еще раз отметить, что сложнофункциональный блок – не только реализация кода, но и полный комплект средств, обеспечивающих контроль правильности функционирования блока и его корректную интеграцию в систему, включая верификационное окружение и наборы функциональных тестов, командные файлы для синтеза, подробную документацию.

1.4. Знакомство с языком Verilog

1.4.1. Введение

Язык описания аппаратуры Verilog появился в 1984 г. Он был создан сотрудником фирмы Gateway Design Automation по имени Phil Moorby для нужд одного единственного проекта. Язык оказался настолько удачным, что был использован и в дальнейшем. В 1989 г. компания Cadence поглотила GDA, в том числе получив права на Verilog-XL. Некоторое время язык оставался внутренним инструментом Cadence, затем в 1990 г. стал доступен для общего пользования. Язык активно применялся и в 1995 г. был стандартизирован как IEEE 1364. Это стандартное подмножество языка до сих пор иногда используется и называется Verilog-95. Синтаксис языка активно развивался, были добавлены многие удобные конст-

рукции, и в 2001 г. стандарт был изменен на IEEE 1364-2001. Это тот стандарт языка Verilog, который используется до сих пор.

Конечно же, процесс развития не стоит на месте. Сейчас активно развивает расширение языка, называемое System Verilog. Оно было стандартизовано в 2005 г. как IEEE 1800-2005. Обратите внимание, что это уже другой стандарт и другой язык.

Этапы развития

1984 – Phil Moorby создал Verilog XL для одного проекта.

1989 – Cadence получает права на Verilog XL.

1990 – язык стал доступен для общего пользования.

1995 – стандарт IEEE 1364.

2001 – стандарт IEEE 1364-2001.

2005 – стандарт System Verilog IEEE 1800-2005.

Несмотря на то, что язык Verilog стандартизован, существует множество его расширений и подмножеств. Каждый производитель САПР с поддержкой Verilog, пытается привнести в него что-то свое, что кажется ему совершенно необходимым и, тем самым, делает написанный код несовместимым с другими САПР. Работая с Verilog, следует всегда соответствовать стандарту языка, так как переносимость кода – абсолютно необходимое требование к описанию любого сложнофункционального блока.

С другой стороны, не любое описание логического устройства на Verilog может быть синтезировано. Каждая система синтеза поддерживает определенное подмножество языка.

1.4.2. Основные конструкции языка

Язык Verilog – очень простой, чем, в первую очередь, и обусловлена его популярность. В языке есть три основные конструкции: комбинационное соединение wire или провод; последовательностный элемент-ячейка reg или регистр; иерархический элемент module или модуль. На базе этих трех базовых элементов – комбинационного, последовательностного и иерархического можно описать любую цифровую схему. Конечно, есть еще масса других языковых конструкций, но все они предназначены для удобства работы с базовыми конструкциями.

Переменные в Verilog могут быть либо битовыми, либо многоразрядными. Последние представляют собой одномерные массивы битовых переменных. Многомерные массивы не поддерживались в языке версии 95. В расширении 2001 г. они предусмотрены, но пользоваться ими следует с большой осторожностью, поскольку далеко не все ПО работает с многомерными массивами корректно.

В Verilog поддерживается логика с четырьмя возможными состояниями. Это логический 0, логическая 1, x – неопределенное состояние и z – высокоимпедансное состояние.

Например:

8-разрядное бинарное число	`8'b01010101; 8'b0`
32-разрядное шестнадцатеричное число	`32'h12345678; 32'hA`
Одноразрядное высокоимпедансное состояние	`1'bz`

Основное отличие языка описания аппаратуры от обычного языка программирования в том, что в Verilog присутствуют понятия времени и события, которое выполняется в определенный момент времени. Событием может быть изменение какого-либо сигнала или группы сигналов, положительный или отрицательный фронт сигнала, а также некоторое специально определенное ключевым словом `event` событие.

Как происходит моделирование. Можно представить себе некоторую ось времени с начальным нулевым значением. В нулевой момент времени исполняются все операции, которые должны произойти при инициализации системы, в первую очередь – инициализация начальных значений переменных.

Любое изменение любой переменной в комбинационном выражении является событием, приводящим к вычислению обновленного значения комбинационного выражения. Если выход комбинационного выражения является входом другого комбинационного выражения, производится вычисление и этого выражения и т.д. Каждое новое вычисление и изменение значений переменных вызывает появление событий, которые, в свою очередь, порождают другие вычисления и т.д.

Если выполнение операции должно произойти с нулевой задержкой, то следующее событие попадает в конец очереди, соответствующей данному моменту времени. Если значение задержки

не равно нулю, то вычисление попадает в очередь, соответствующую моменту времени, который смещен относительно текущего момента на время задержки.

Каждому моменту времени соответствует очередь вычислений. В пределах одной очереди все вычисления производятся в один и тот же момент времени и, если между ними нет прямой взаимосвязи, считают, что все они выполняются параллельно. Стандарт не определяет, какое из вычислений случится раньше или позже.

Например:

```
initial    A = 1;
initial    B = A + 2;
```

Эти два присваивания исполняются в один и тот же момент времени моделирования, вне зависимости от порядка их написания. Какое из них выполнится раньше, зависит от реализации в конкретной версии симулятора.

Для последовательностного элемента вычисление и запись значения производится по заданному событию. В отсутствии этого события, вне зависимости от изменения входных переменных, значение регистра сохраняется. Типичное событие, появление которого вызывает запись значения в регистр, – изменение логического состояния, т.е. фронт тактового сигнала.

Например:

```
reg R;
always @(posedge CLK)
   R = A & B;
```

В данном случае событие – положительный фронт тактового сигнала `CLK`, по которому в регистр `R` осуществляется запись значения `A & B`.

Для иерархического элемента изменение значения на входе или выходе порождает операцию переноса изменившегося значения через иерархическую границу.

Когда все операции из очереди вычислений для данного момента времени выполнены, процесс моделирования переходит к следующему дискретному моменту времени, которому соответствует

не пустая очередь операций. Шаг моделирования определяется минимальной задержкой вычислений. Останов моделирования производится по специальной системной команде `$finish` в заданный момент времени.

Следует всегда помнить, что любая операция в языке Verilog происходит в строго определенный момент времени. Несколько операций, соответствующих одному и тому же моменту времени, могут исполняться в разном порядке, если их взаимосвязь не определена однозначно. Примеры, иллюстрирующие порядок выполнения операций, приведены в табл. 1. В левом столбце значение B вычисляется после того, как вычислено значение A. В правом столбце изменение A не влияет на результат вычисления B.

Таблица 1

Исполняются последовательно	Исполняются параллельно

Было A = 0; B = 0;
`begin`
`A = 1;`
`B = A + 1;`
`end`
стало A = 1; B = 2;
	Было A = 0; B = 0;
	`initial A = 1;`
	`initial B = A + 1;`
	
	стало A = 1; B = 1;

Было A = 0; B = 0;
`initial #1 A = 1;`
`initial #2 B = A + 1;`
 	
стало A = 1; B = 2;
	Было A = 0; B = 0;
	`fork`
	`A = 1;`
	`B = A + 1;`
	`join`
	стало A = 1; B = 1;

Следует четко различать операции по вычислению комбинационных значений и последовательностных элементов. Комбинационные элементы вычисляются сразу же при изменении любого из входных воздействий, что соответствует аппаратной комбинационной схеме. Последовательностные элементы, т.е. регистры вычисляются только по заданному определенному событию и сохраняют свое значение до следующего события.

В Verilog определены два вида операций присваивания значений для регистров – блокирующее и неблокирующее присваивания. Блокирующее присваивание исполняется сразу в том порядке, в котором оно описано в коде. Все неблокирующие присваивания, относящиеся к одному и тому же событию, исполняются параллельно.

Таким образом, входными значениями всех переменных в выражениях являются значения, соответствующие моменту времени до события, вызвавшего запись в регистры. После наступления события во всех регистрах одновременно появляются новые значения. Такой механизм наиболее точно отражает функционирование реальных последовательностных элементов. Соответствующие примеры приведены в табл. 2.

Таблица 2

Блокирующее присваивание	Неблокирующее присваивание

Было A = 0; B = 0;
`always @(posedge CLK)`
`begin`
`A = 1;`
`B = A + 1;`
`end`
стало A = 1; B = 2;
	Было A = 0; B = 0;
	`always @(posedge CLK)`
	`begin`
	`A <= 1;`
	`B <= A + 1;`
	`end`
	стало A = 1; B = 1;

Было A = 0; B = 1;
`always @(posedge CLK)`
`begin`
`A = B;`
`B = A;`
`end`
стало A = 1; B = 1;
	Было A = 0; B = 1;
	`always @(posedge CLK)`
	`begin`
	`A <= B;`
	`B <= A;`
	`end`
	стало A = 1; B = 0;

1.4.3. Описание комбинационных блоков

Комбинационные переменные могут быть описаны как переменные типа `wire`. Комбинационными переменными могут быть также выходы блока – `output`.

Например:

```
wire [7:0] N;       // 8-разрядная шина N
```

Вычисление комбинационной переменной может быть описано непосредственно при объявлении переменной:

```
wire [7:0] N = 8'b00000001;
wire [7:0] M = N & 8'h0A;
```

Можно также использовать оператор комбинационного присваивания assign:

```
assign M[7:0] = N & 8'h0a;
```

Если производимые вычисления – сложнее, чем можно записать одно строкой, проще всего и нагляднее использовать описание вычислений в виде функции:

```
function [7:0] AND8;
   input [7:0] a;
   input [7:0] b;
   begin
      AND8 = a & b;
   end
endfunction
assign M = AND8(N, 8'h0a);
```

Можно также использовать конструкцию `always`, задав в качестве возможных событий изменения всех входных сигналов. Комбинационную переменную при этом придется описать типом `reg`. Такое описание является нежелательным, так как оно является источником дополнительных ошибок.

Описывая комбинационные цепи, следует обратить особое внимание на оператор выбора:

```
assign D = C ? A : B;
// Если C не равно 0, то D = A, в противном случае D = B
```

Этот оператор редко используется в обычных языках программирования, но широко применяется в языке Verilog, так как непо-

средственно описывает мультиплексор без использования конструкций ветвления.

1.4.4. Описание последовательностных блоков

Как уже упоминалось, принципиальным отличием последовательностных элементов типа reg от комбинационных элементов является то, что запись в них производится по определенному событию. Для описания элемента это событие нужно указать. Используется ключевое слово always, после которого указывается необходимое событие. Чаще всего это появление положительного фронта тактового сигнала.

Событий может быть несколько. Например, положительный фронт тактового сигнала и отрицательный фронт сигнала сброса. Изменение любого из них может привести к изменению значения регистра:

```
always @(posedge CLK or negedge RST)
   if (RST == 1'b0)   //Если сигнал сброса нулевой (активный)
      D <= 1'b0;
   else      //Если сигнал сброса не нулевой, событие – фронт CLK
      D <= A & B;
```

1.4.5. Описание иерархических блоков

Описание иерархического блока помещается между служебными словами module и endmodule. Каждый модуль имеет уникальное имя, характеризующее его тип. Модуль может использоваться в другом модуле в качестве структурного элемента. Допустимо использовать несколько экземпляров, причем, каждый экземпляр модуля представляет собой отдельный объект с вполне определенным именем, подключением выводов, параметрами и т.д.

Модуль имеет набор интерфейсных сигналов ввода-вывода, через которые осуществляется обмен информацией с остальной схемой.

Например:

```
module AND3 (X, A, B, C);
   input A, B, C;
```

```
    output X;
    assign X = A & B & C;
endmodule
```

Вызов модуля:

```
AND3 AND3_00( .X(xout), .A(ain), .B(bin), .C(cin));
```

С точкой впереди указываются имена сигналов вызываемого модуля. В скобках описываются сигналы модуля вызывающего.

Не все устройства, которые можно описать на языке Verilog, могут существовать в действительности. Используя, например, нулевые задержки сигнала или бесконечные циклы можно описать нереализуемую конструкцию. Подобные трюки возможны лишь при описании среды моделирования и тестового окружения модели. Там все средства хороши, поскольку ни среда, ни тестовое окружение не входят в состав проектируемой СнК. Вместе с тем все блоки микросхемы должны быть реализуемы.

Что это означает? В первую очередь, что не всякий код может быть синтезирован. Синтезаторы поддерживают только такое подмножество языка Verilog, которое с большой вероятностью обеспечивает создание синтезируемой схемы. Точную информацию о подмножестве языка Verilog, которое поддерживает используемая программа синтеза, можно узнать из пользовательской документации.

В частности, не поддерживаются такие конструкции, как события – event. Поэтому необходимо использовать обычные сигналы для передачи управляющей информации. Не поддерживаются абсолютные задержки, так как их невозможно точно задать и в реальной схеме. Нет возможности использовать блоки initial, так как реально в системе нет нулевого момента старта и для задания начального состояния регистров нужно использовать сигнал сброса. Для того чтобы упростить работу синтезатора и гарантировать получение ожидаемого результата, рекомендуется использовать только типовые конструкции при описании логических элементов, такие, например, как приведенные ранее для регистров.

При описании мультиплексоров, как правило, используется оператор выбора. Для многовходовых мультиплексоров применяется конструкция case, причем, если в конструкции case определены все

возможные комбинации сигналов, то будет порождена комбинационная схема, в противном случае – набор регистров:

```
case (sel[1:0])
    2'b00: A = 1'b0;
    2'b01: A = B;
    2'b10: A = ~B;
    default: A = 1'b1; // все остальные случаи
endcase
```

Строгость требований к синтезируемому описанию изменяется по мере развития программ синтеза электрических схем. Новые программы поддерживают более совершенные алгоритмы синтеза, что снижает требования к тщательности написания кода. Нет необходимости оптимизировать код вручную, если с этим гораздо лучше справится автомат. Современная тенденция в написании синтезируемого кода – писать не столько оптимальный, сколько читаемый и переносимый код.

Современные программы синтеза обеспечивают поддержку многих сложных схемотехнических конструкций в автоматическом режиме. Также для разработчиков отпала необходимость в проектировании и оптимизации таких вычислительных блоков, как сумматоры, умножители, многооперандные сумматоры. Достаточно указать в тексте модели обозначение необходимой операции, позаботиться о правильности форматов операндов, и САПР подберет подходящий вычислительный блок в автоматическом режиме. Набор готовых функциональных блоков, конечно же, не ограничивается только сумматорами и умножителями. Он гораздо шире. Если нужно использовать какой-то библиотечный блок, функцию которого не получается описать стандартными операторами, например вычисление синуса, такой блок вручную добавляется в код в качестве иерархического элемента.

Язык Verilog используется не только для описания RTL кода, но и для описания функционирования библиотечных элементов и принципиальных электрических схем. С этой целью применяются дополнительные конструкции языка Verilog, которые представляют собой те же модули и которые, следовательно, можно использовать

в построении иерархии логической схемы. Функциональность элемента, как правило, характеризуется при помощи стандартных логических примитивов: буфера buf, инвертора inv, логических элементов and, or, xor. Регистры описываются в виде таблицы состояний.

Библиотечные элементы могут строиться по иерархическому принципу, на основе базовых примитивов. Характерной составляющей библиотечного элемента является блок specify. В нем определяются параметры элементов, в первую очередь временные. Это могут быть задержки в элементах от входов до выходов, требования к минимальной длительности интервалов времени предварительной установки (предустановки) и постудержания сигнала для регистров и т.д.

Следует заметить, что в самом библиотечном элементе определяется только состав необходимых параметров. Значения параметров при этом являются приблизительными или вовсе условными. Все дело в том, что невозможно определить параметры для какого-то типа элемента, поскольку они являются индивидуальными для каждого экземпляра элемента, использованного в схеме.

Также при помощи языка Verilog производится описание синтезированной электрической схемы. Отличие от RTL описания в том, что в схемном описании полностью отсутствуют логические конструкции. Схема представляет собой иерархическую структуру соединенных между собой модулей. Модули нижнего иерархического уровня – библиотечные элементы. Можно сказать, что в таком представлении электрическая схема – это не что иное, как описание связей между набором библиотечных элементов. Упомянутый ранее термин «нетлист» так схему и характеризует – список связей или, точнее, цепей, т.е. список элементов и связей между ними.

Рассмотрим в качестве примера описание на Verilog четырехразрядного двоичного сумматора с последовательным переносом. Схема устройства представлена на рис. 2, где A[3:0] и B[3:0] – входные слова, S[3:0] – сумма, AF1 – одноразрядный сумматор.

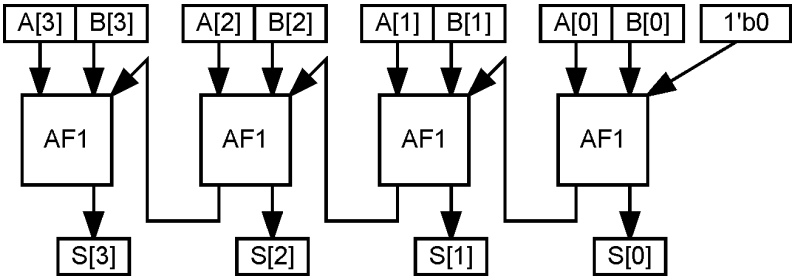


Рис. 2. Структурная схема 4-разрядного сумматора

Устройство можно описать различными способами. Например, можно использовать логические выражения. Это классический RTL код:

```
module ADD4(S, A, B);
output [3:0] S;
input [3:0] A, B;
wire [3:0] P;
assign P[3:0] = A[3:0] & B[3:0] | A[3:0] &
   {P[2:0],1'b0} | {P[2:0],1'b0} & B[3:0];
assign S[3:0] = A[3:0] ^ B[3:0] ^ {P[2:0],1'b0};
endmodule
```

Можно описать отдельный элемент – одноразрядный сумматор и построить устройство на его основе:

```
   ADD1 a0(.S(S[0]), .P(P[0]), .A(A[0]), .B(B[0]),
.C(1'b0));
   ADD1 a1(.S(S[1]), .P(P[1]), .A(A[1]), .B(B[1]),
.C(P[0]);
   ADD1 a2(.S(S[2]), .P(P[2]), .A(A[2]), .B(B[2]),
.C(P[1]);
   ADD1 a3(.S(S[3]), .P(P[3]), .A(A[3]), .B(B[3]),
.C(P[2]);
```

Если одноразрядный сумматор является библиотечным элементом, реализованным для заданной технологической библиотеки, то показанный текст – типичный нетлист.

Возможно и другое описание. Так как синтезатор электрической схемы легко справится с операцией суммирования, устройство можно описать, используя арифметическое выражение:

```
assign S = A + B.
```

Выполнив логический синтез на основе этого кода, получим схему из предыдущего примера.

1.5. Создание модели устройства на языке Verilog

Рассмотрим пример построения модели устройства. Создадим модель четырехразрядного счетчика с асинхронным сбросом. Структурная схема и временная диаграмма работы счетчика показаны на рис. 3, где RG – четырехразрядный регистр; INC – инкрементор (увеличивает входной код на 1); RST – сигнал асинхронной установки в 0 (сброс); CLK – тактовый сигнал; Q – выходной код.

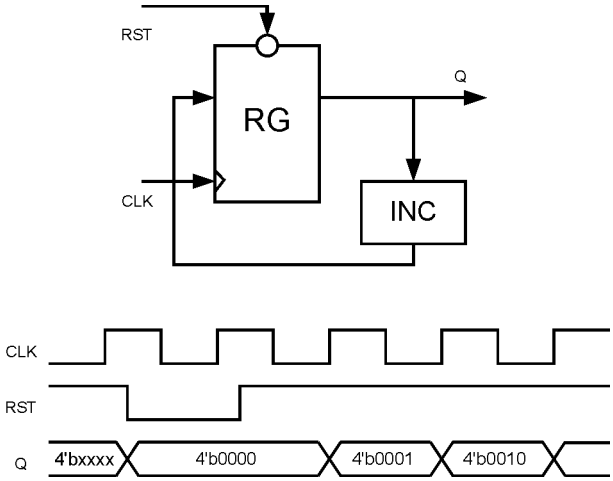


Рис. 3. Структурная схема 4-разрядного счетчика
и временные диаграммы его работы

Модель устройства представляет собой модуль, содержащий четырехразрядный регистр с асинхронным сбросом и комбинационную схему инкрементора:

```
module CNT4(Q, RST, CLK);
output [3:0] Q;
input RST, CLK;
reg [3:0] Q;
always @(posedge CLK or negedge RST)
   if (RST == 1'b0)
      Q <= 4'b0; // Начальная установка - сброс
   else
      Q <= Q + 1'b1; // Инкремент счетчика
endmodule
```

Для того чтобы проверить правильность функционирования модели, необходимо создать тестовое окружение моделируемого устройства. Тестовое окружение – иерархический модуль на языке Verilog, который представляет собой модель всего, что может окружать разрабатываемое устройство. Целью построения тестового окружения является всесторонняя проверка модели на правильность функционирования, разработка и проверка тестов, которые будут применяться на других этапах разработки устройства, и отладка разрабатываемой модели.

Сложность разрабатываемых устройств может быть очень высокой. Мало просто проверить функционирует модель или нет. Если ее функционирование не соответствует ожидаемому поведению, то нужно обеспечить возможность локализации ошибок, чтобы их можно было обнаружить и исправить.

Что включается в модель тестового окружения? Очевидно, что необходимо задать сигналы на входах устройства, т.е. описать входные переменные. Если их значения определяются непосредственно в модуле тестового окружения, то переменные будут регистрами – ячейками, значения которых можно определить и переопределить.

Входные сигналы могут формироваться при помощи различных конструкций и логических выражений. Например, так обычно описывается генератор тактового сигнала:

```
reg CLK = 1'b0;
initial
   forever #10 CLK = ~CLK;
```

Для создания сложных последовательностей могут использоваться подпрограммы (task). Они принципиально отличаются от функций (function) тем, что в их описании присутствует такой параметр, как время, и могут указываться задержки между отдельными операциями, т.е. можно описать продолжительный по времени набор изменяющихся значений.

Нужно следить за тем, чтобы одна и та же подпрограмма task не вызывалась заново до того, как завершится предыдущий вызов, так как это приведет к появлению непредсказуемых значений переменных. Например, если продолжительность исполнения подпрограммы составляет три процессорных такта, не следует вызывать ее каждый такт. Это иллюстрируется следующим фрагментом, содержащим описание подпрограммы и ее вызов:

```
// Описание подпрограммы
task sgen;
begin
   a = 1;
   @(posedge CLK) a = 3;
   @(posedge CLK) a = 7;
end
endtask


// Вызов подпрограммы
#1 sgen; // Вызов процедуры, длительность ее исполнения 3 такта
@(posedge CLK) sgen;
// Вызов процедуры через 1 такт приведет к непредсказуемому результату
```

```
repeat (3) @(posedge CLK); sgen;
// Вызов процедуры через три такта после предыдущего корректен

@(posedge CLK) sgen1;
// Вызов другой процедуры возможен в любой момент времени
```

Значения переменных на входах модели некоторого устройства могут устанавливаться путем подключения к нему модели другого устройства.

Если разработаны тесты функционального контроля для какого-либо блока, то входные переменные могут быть заданы в виде тестовых векторов – наборов значений, определенных для всех входов устройства и подаваемых с некоторой частотой, например одновременно с каждым фронтом сигнала синхронизации.

Кроме подачи входных сигналов, конечно же, нужно осуществлять контроль получаемых результатов – значений сигналов на выходах устройства.

Возможны разные варианты контроля выходных значений. Самый простой способ – визуальный, он применим для устройств небольшой сложности. В этом случае результаты моделирования выводятся в виде текстовых таблиц (дампов) или временных диаграмм. Форматы вывода данных различаются в различных САПР, однако есть и унифицированные форматы, например VCD (Value Change Dump). Отображение временных диаграмм на основе информации из файлов этого формата поддерживается многими программными средствами различных производителей.

В случае сложного устройства простой визуальной проверкой проконтролировать правильность функционирования устройства невозможно. В этом случае необходимо использовать автоматические методы контроля. Основными методами автоматического контроля являются проверки на выполнение формальных правил и использование эталонных моделей.

Формальные правила (assertions) могут задаваться разработчиком в процессе написания кода модели. Они призваны отслеживать появление исключительных ситуаций в процессе работы устройства. Например, недопустимо одновременное появление активных сигналов асинхронного сброса и установки регистра. Возникновение подобной ситуации может быть выявлено в процессе проверки формальных правил.

Язык Verilog задание и контроль формальных правил не поддерживает. Они являются составной частью языка нового поколения – SystemVerilog, или могут являться функциональным расширением Verilog в конкретной реализации САПР. Формальные правила не обеспечивают полной проверки правильности функционирования модели, но могут ускорить отладку устройства.

Вторым и самым эффективным методом автоматического контроля является использование эталонных моделей устройств. На тестируемую и эталонную модели устройства подают одни и те же входные данные и результаты работы сравниваются. Таким способом можно получить однозначный ответ о правильности функционирования моделей для большого набора тестов.

Существуют разные виды эталонных моделей. Это может быть поведенческая модель на Verilog или SystemVerilog, функционирующая в составе модели тестового окружения. Можно создать модель на языке SystemC или C++. Она также может быть встроена в тестовое окружение. Тогда эталонное моделирование проводится одновременно с основным. Можно выполнить эталонное моделирование отдельно от основного с сохранением результатов. В этом случае в модели тестового окружения осуществляется загрузка результатов эталонного моделирования и сравнение их с данными от рабочей модели. Если наблюдается расхождение результатов моделирования, проверяется совпадение данных в специально заданных внутренних точках, что позволяет локализовать источник ошибок.

Тестовое окружение создается как для отладки отдельных блоков в процессе их разработки, так и для микросхемы в целом. Тестовое окружение всей микросхемы используется, кроме прочего, для разработки и верификации тестов функционального контроля. Они используются в дальнейшем для верификации изготовленных микросхем.

1.6. Логический и физический синтез схем

Логический синтез электрических схем представляет собой перенос технологически независимого описания устройства – RTL кода, в заданный технологический базис. Синтез сопровождается оптимизацией схемы для достижения заданных параметров.

Обычно логический синтез включает следующие этапы: входной анализ кода модели, трансформирование кода в схему на базе абстрактных логических элементов (elaborate), загрузку файла с целевыми параметрами схемы, логическую оптимизацию схемы, замену абстрактных элементов на элементы из заданной технологической библиотеки (mapping), оптимизацию полученной схемы, дополнительную обработку полученного результата.

Инструкции синтезатору, определяющие параметры отдельных этапов синтеза, объединяются в командном файле, который управляет процессом создания схемы.

Входной анализ кода – формальная проверка синтаксиса RTL кода на соответствие его подмножеству, поддерживаемому данной программой синтеза. Как уже отмечалось, синтезаторы, обычно, поддерживают только некоторое подмножество языка, которое легко может быть преобразовано в электрическую схему. Конструкции, которые синтезатором не поддерживаются, игнорируются, если это возможно, или вызывают сообщение об ошибке.

Синтезируемые подмножества языка Verilog могут отличаться для различных программ синтеза и их версий. Со временем эта поддержка расширяется. Современные САПР компаний Cadence и Synopsys поддерживают большую часть конструкций Verilog, Verilog-2001, а также SystemVerilog. В дополнение к стандартному синтаксису ряд САПР использует нестандартизованные расширения, которые предназначены для того, чтобы облегчить жизнь разработчикам.

Например, специальной директивой в строке комментария:

```
A <= 0;
//synopsys translate_off
B <= A; // Эта строка будет исключена из входного кода
//synopsys translate_on
```

можно отключить анализ части кода. Эти строки будут проигнорированы программой синтеза, но будут учтены в процессе моделирования RTL кода.

Входной анализ кода производится для всех файлов проекта. Дальнейшая работа программы синтеза возможна только при успешно завершенном входном анализе.

Следующим этапом является преобразование RTL кода в схему на базе абстрактных логических элементов. Этот базис включает в себя семейство стандартных логических вентилей, а также мультиплексоры и универсальные последовательностные элементы с асинхронной установкой и сбросом. Описание в виде RTL кода транслируется в схему, состоящую из этих элементов, так, чтобы логика работы устройства при этом не изменилась. В англоязычной литературе этот процесс обычно называют elaboration. В программе синтеза есть команда с тем же названием.

Как только получена схема, ее можно проанализировать и получить первые, еще предварительные оценки параметров проектируемого устройства. На данном этапе уже можно посмотреть структуру проекта, иерархию модулей и выявить потенциальные ошибки. Например, получить список выводов, которые по тем или иным причинам оказались не присоединенными.

На этом же этапе к схеме присоединяются макроэлементы из библиотеки программы синтеза. Если, например, в тексте RTL описания встретится операция суммирования, то в нужное место схемы будет включен библиотечный элемент – сумматор. Это отдельный блок, состоящий из абстрактных логических элементов.

Прежде чем приступить к оптимизации полученной схемы, нужно задать необходимые параметры-ограничения (constraints). Они могут быть временными и конструктивными, включая топологические параметры при наличии этапа физического синтеза. В частности, могут быть заданы целевые значения предполагаемой площади блока и его составных частей.

Временные параметры являются основными при определении работоспособности микросхемы.

Большинство цифровых схем относятся к синхронным устройствам. Они должны функционировать на заданной частоте без сбоев, а это означает, что за время одного процессорного такта сигнал должен успеть пройти от входа до выхода комбинационной схемы по всем возможным путям. Варианты маршрутов прохождения сигнала показаны на рис. 4.

В файле параметров отмечаются все тактовые сигналы в схеме с указанием их частоты и возможного разброса параметров, например:

```
create_clock [get_ports {CLK}] -period 3000 -waveform {0
1500} # Описание тактового сигнала

set_clock_uncertainty 1000 [get_clocks {CLK}]
# Возможный разброс частоты тактового сигнала
```

Находятся задержки на входах и выходах устройства. Определяются все возможные исключения – реально не существующие пути, пути, которые сигнал может проходить за время более одного такта и т. д.

Временные параметры устройства считаются полностью определенными, если заданы характеристики тактовых сигналов для всех регистров и установлены ограничения на максимальную задержку по всем возможным путям.

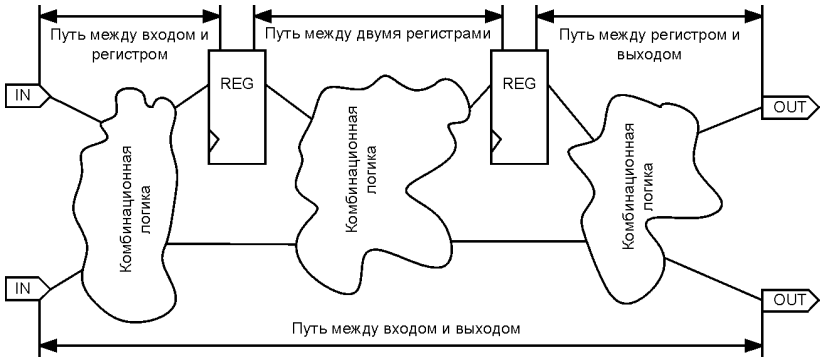


Рис. 4. Пути прохождения сигналов в цифровой схеме

Кроме требований к задержкам могут быть также определены требования к максимальным длительностям фронтов сигналов. Этот параметр непосредственно связан с такими конструктивными параметрами блока, как предельно допустимое число нагрузок и максимальная емкость на выходах элементов.

Все параметры могут быть заданы либо в командном файле, который управляет процессом синтеза, либо в отдельном файле формата .sdc. Этот формат, разработанный фирмой Synopsys, является, де-факто, стандартом в области записи параметров устройства для синтеза и может быть использован в различных системах проектирования. Он всегда включается в комплект поставки СФ блоков.

Кроме рассмотренных параметров, могут задаваться и другие, например ограничения на потребляемую мощность блока. Такие параметры являются специфичными для определенной реализации САПР, они не включаются в файл формата .sdc, а задаются непосредственно в командном файле.

Следующий этап – логическая оптимизация схемы. Она производится до того, как определена целевая технология, поэтому является предварительной и выполняется по формальным критериям. На этом этапе производится удаление избыточных элементов, например регистров, выходное значение которых не используется; оптимизация комбинационной логики, константных портов и т.д. Кроме того, могут быть произведены дополнительные операции вручную, например преобразование иерархии системы, декомпозиция структур и создание новых иерархических блоков или, напротив, группировка нескольких блоков в один блок.

Одним из основных этапов синтеза является перенос схемы в заданный технологический базис (mapping). На этом этапе абстрактные логические элементы заменяются элементами из заданной технологической библиотеки. Замена может производиться как один в один, так и группами, так как библиотечные элементы могут быть сложнофункциональными.

Библиотека элементов представлена в виде одного или нескольких библиотечных файлов. Общеупотребительным форматом является .lib – так называемый liberty формат. Это описание библиотечных элементов в текстовом виде, которое содержит следующие данные: функциональное назначение и структура, параметры элементов и функции зависимости этих параметров от изменения внешнего окружения, в частности от изменения емкости соединений. Кроме того, библиотека включает набор технологических параметров, например емкостей связей в зависимости от размеров блоков, предельных значений нагрузок на одну связь, максимальнодопустимой длительности фронтов сигналов и т.д. Описание в виде текстового файла не всегда удобно, так как дает полный доступ к технологической информации и поэтому с целью защиты данных, практически, все производители программ синтеза поддерживают бинарные форматы библиотек элементов несовместимые друг с другом. После завершения данного этапа получается электрическая схема, но еще в неоптимизированном виде.

Основным этапом синтеза является оптимизация. В процессе ее электрическая схема приводится в соответствие с определенными для нее параметрами. На основе заданных параметров строится целевая функция оптимизации. Вычисляется ее значение. Для определения значений временных параметров используется статический временной анализатор. Можно сказать, что статический временной анализатор – это ключевой элемент программы оптимизации. В зависимости от того, насколько точно удастся определить временные параметры системы, зависит корректность работы микросхемы.

Часто критически важно быть уверенным, что система будет работоспособна на заданной частоте. Если максимальную частоту можно оценить с точностью не более 50 %, то это может привести к значительному перерасходу аппаратных ресурсов и возрастанию стоимости микросхемы. Например, оценки максимальной частоты работы системы дают ожидаемые значения в диапазоне от 100 до 200 МГц, тогда в расчете на наихудший случай придется принять этот показатель равным 100 МГц, хотя реальное значение может оказаться лучше. Если же точность определения параметров будет еще ниже, то получение работоспособных БИС становится проблематичным.

Для того чтобы определить временные параметры устройства, нужны достаточно точные физические модели элементов, адекватные используемой технологии. Ключевым параметром в статическом временном анализе является емкость связи на выходе элемента, которая определяет длительность задержки. Кроме того, необходимо учитывать задержки на трассах межсоединений, которые зависят от длины связей.

Конечно, когда топология микросхемы уже разработана, рассчитать длины связей между элементами не является большой проблемой. Но что делать, если выполняется логический синтез схемы, а никакой топологии еще нет? В таком случае применяют методы приближенного расчета длины межсоединений. На рис. 5 проиллюстрированы три наиболее часто используемых метода.

Первый из них основан на использовании таблиц длин связей. В этом случае расчет производится исходя из размеров блока, в котором проходит трасса. По известной величине площади блока его размеры оценивают, полагая, что форма блока близка к квадратной.

Длину связи определяют как длину пути между двумя наиболее удаленными точками блока. Это хорошая оценка сверху.

Метод давал удовлетворительные результаты при оценке параметров схем, реализуемых по технологиям с проектными нормами до 0,25 мкм. С уменьшением проектных норм его точность падает, так как погрешность в оценке длин связей начинает в большей степени сказываться на достоверности результатов оценки параметров схем. Для сохранения точности требуется уменьшать шаг представления данных в таблице возможных значений размеров блоков и длин связей. Кроме этого, недостатком метода является предположение о квадратной форме блоков. Как только форма блоков начинает отличаться от квадратной, уменьшается точность оценки временных параметров.

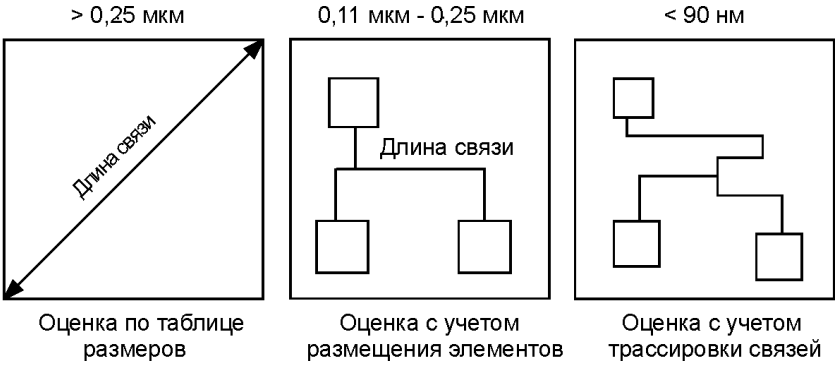


Рис. 5. Оценка длин связей в схеме

Для технологий с проектными нормами 0,25 мкм и ниже вклад элементов в суммарную задержку становится меньше, чем вклад внешних связей. С уменьшением проектных норм эта тенденция проявляется сильнее и, следовательно, влияние внешних связей необходимо оценивать точнее.

Решением вопроса стало добавление этапа так называемого физического синтеза. На этом этапе создается и электрическая схема, и топология блока. Выполняется и оптимизируется размещение библиотечных элементов. Зная точное размещение элементов, можно гораздо точнее оценить длины связей между ними.

Для субмикронных технологий с проектными нормами меньше 130 нм и такого приближения уже недостаточно. Нужно знать не просто длину проводника, а его реальное расположение, поскольку он может проходить в нескольких слоях, имея там разную ширину и, соответственно, паразитную емкость. В этом случае размещением элементов не ограничиваются, а производят полную трассировку связей. Конечно, вычисление задержек в схеме, основанное на данных, полученных в результате предварительного размещения и трассировки элементов, – задача, требующая больших вычислительных затрат.

Время задержки сигнала на пути между двумя регистрами включает задержку регистра-источника, суммарную задержку комбинационных элементов и связей между ними, а также время предустановки регистра-приемника. Кроме того, надо учитывать задержку, связанную со случайным разбросом длительности периода тактового сигнала (рис. 6).

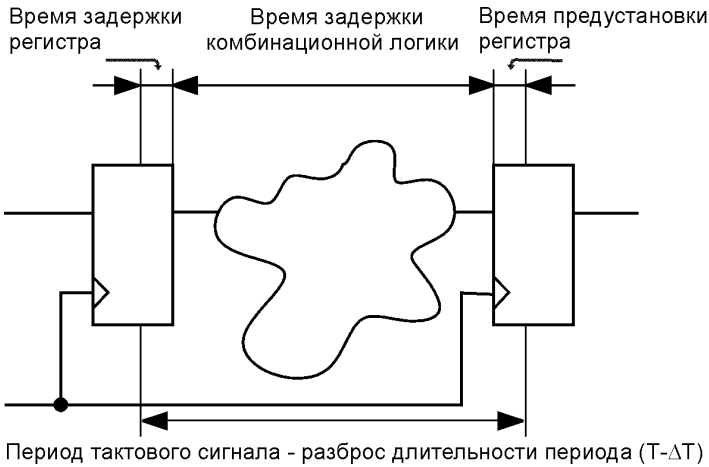


Рис. 6. Вычисление задержки сигнала в схеме с регистрами

Значения задержек обычно являются определяющими в целевой функции оптимизации. Сама оптимизация заключается в подборе наиболее подходящих логических элементов путем эквивалентной замены. Подбирается также мощность буферов на выходах элементов, определяющая их нагрузочную способность – максимальное

число нагрузок, которое можно подключить к выходам без существенного ухудшения временных параметров. При этом учитывается, что более мощные элементы имеют большее энергопотребление и занимают большую площадь. На этапе физического синтеза осуществляется также оптимизация размещения элементов и оптимизация трассировки. Алгоритмы оптимизации – важнейшая часть программ синтеза. Они не раскрываются и являются наиболее ценной собственностью компаний – разработчиков САПР.

Вычисление параметров схемы, в первую очередь временных, производится для определенных условий ее работы. Существует специальный термин – PVT (Process-Voltage-Temperature), характеризующий область в некотором трехмерном пространстве параметров, в пределах которой устройство должно оставаться работоспособным.

Положение рабочей области определяется особенностями технологического процесса – возможным разбросом параметров элементов (на одной пластине и на разных пластинах), а также напряжением питания и температурой окружающей среды. Для расчета различных параметров используются различные сочетания условий. Так, задержки элементов будут максимальными при минимальном напряжении питания и максимальной температуре. Для минимизации задержек нужны противоположные условия. С другой стороны, чтобы оценить максимальную потребляемую устройством мощность, необходимо рассмотреть вариант максимального напряжения питания и максимальной температуры окружающей среды. Для различных вариантов сочетания условий работы устройства создаются отдельные библиотеки с различными параметрами элементов. Обычно синтез выполняют с вычислением максимальных задержек, а затем производят верификацию проекта для минимальных задержек.

Современные САПР позволяют синтезировать цифровые устройства, содержащие несколько сотен тысяч логических вентилей. Соответствующий предел определяется мощностью используемых рабочих станций.

А что делать, если число элементов в СнК превышает этот предел? Возможны два подхода к синтезу схем – восходящее и нисходящее проектирование.

Методология нисходящего проектирования предполагает синтез всей схемы целиком. Она позволяет получить наилучший результат с наименьшими затратами, но размер схем, которые можно синтезировать таким способом, ограничен возможностями современных вычислительных систем, на которых работает САПР.

Другой подход – восходящее проектирование, при котором компоненты системы синтезируются по отдельности. Главная проблема, которую в таком случае приходится решать, – правильность описания взаимодействия между компонентами. Может, например, оказаться, что путь распространения сигнала проходит от регистра одного блока к регистру другого блока, который синтезируется отдельно от первого. Для получения правильного результирующего значения задержки необходимо корректно разделить задержки частей пути в каждом блоке, как показано на рис. 7.

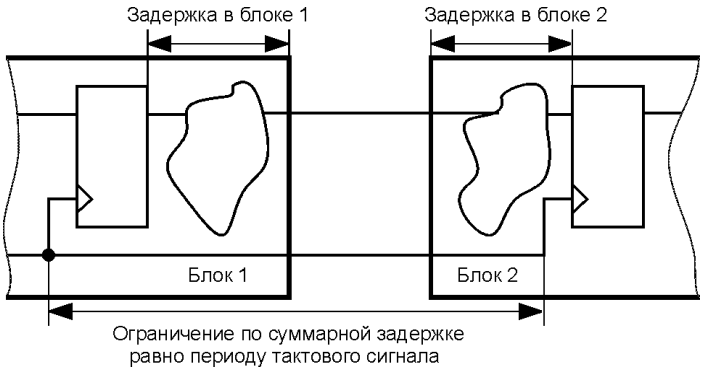


Рис. 7. Распределение времени задержки между двумя блоками

Такой процесс называется budgeting. Он может выполняться в автоматическом режиме. При этом в синтезатор загружается вся схема, задаются параметры синтеза на уровне системы, а программа генерирует параметры синтеза для отдельных блоков. Далее с использованием этих параметров блоки могут быть синтезированы по отдельности.

Если выполняется физический синтез, то для получения параметров синтеза отдельного блока используется еще план кристалла,

позволяющий провести оценку взаимодействия этого блока с другими устройствами системы, учитывая их взаимное расположение.

Получение параметров синтеза для отдельных блоков – сложная задача, не имеющая однозначного решения. В СнК может использоваться множество блоков разных разработчиков. Гарантировать корректное соединение блоков с соблюдением необходимых временных соотношений в этом случае можно лишь одним способом – развязывая пути между блоками регистрами. Хорошей можно считать схему, в которой все выходы являются регистровыми.

Финальной стадией синтеза является так называемая оптимизация по месту (inplace optimization). Она выполняется, когда уже готова топология чипа, и позволяет вносить в схему лишь незначительные исправления, например изменять мощность буферов с целью коррекции временных параметров или снижения энергопотребления.

Следует отметить, что период смены поколений программ синтеза очень небольшой (2–3 года). Как правило, появление новой технологии приводит к тому, что имеющиеся программы синтеза оказываются уже недостаточно точными или производительными и требуется их существенное обновление.

1.7. Организация тестирования схем

Разрабатываемую СнК необходимо проверять на всех этапах разработки. Есть два основных вида контроля – функциональный и аппаратный.

Функциональные тесты предназначены для проверки того, что система на кристалле правильно выполняет все необходимые задачи. Тесты функционального контроля создаются на этапе разработки модели системы и используются далее на всех этапах проектирования. Кроме, собственно, функционального контроля функциональные тесты используются для проверки целостности системы, т.е. для того, чтобы убедиться, что на определенном этапе разработки, например при синтезе схемы, система не потеряла и не изменила заданную функциональность.

Кроме того, за сохранением целостности системы можно следить, используя программы формальной логической верификации. Они проверяют сохранение логики работы системы по формаль-

ным логическим выражениям. Такие инструменты незаменимы при сравнении различных версий электрических схем.

Вместе с тем они могут оказаться излишне строгими при контроле преобразования RTL кода в электрическую схему, если, например, RTL код содержит избыточную логику, удаляемую программой синтеза.

Обычной практикой является разделение разработки RTL кода и тестов между разными разработчиками и даже компаниями. Таким способом достигается лучшее качество тестирования.

Качество тестов можно проверить при помощи специализированного программного обеспечения. Типичными показателями качества функциональных тестов является покрытие тестами RTL кода и покрытие переключений на выводах модулей. Каждое логическое выражение в коде должно тестироваться и каждый вывод модуля должен изменить свое логическое состояние в обоих направлениях, т.е. переключиться из "1" в "0", а также из "0" в "1".

Тесты аппаратного контроля предназначены для выявления дефектов, возникающих при производстве микросхем. В отличие от функциональных тестов основной задачей аппаратных тестов является проверка правильности всех связей в микросхеме, отсутствие разрывов дорожек и коротких замыканий цепей. Для того чтобы осуществить подобную проверку, необходимо последовательно изменять логические состояния на выходах всех логических элементов схемы.

В случае относительно простых схем этого можно добиться путем написания достаточного числа функциональных тестов. До недавнего времени, пока микросхемы были еще не слишком большими, функциональные тесты использовались также и для тестирования самих микросхем. Для этого они преобразовывались в наборы так называемых тестовых векторов, представляющих собой коды на всех входах и выходах микросхемы.

Тестовые воздействия последовательно, вектор за вектором, подавались на входы микросхемы, а коды на выходах сравнивались с эталонными значениями. В результате тестирования принималось решение о функциональной пригодности микросхемы.

В случае таких сложных систем, как СНК, аппаратное тестирование с использованием функциональных тестов невозможно. Наличие очень большого числа логических устройств приводит к то-

му, что обеспечить смену состояний на выходах некоторых элементов становится очень сложным или вовсе невозможным.

Для обеспечения тестируемости современных схем придуман метод, обеспечивающий доставку тестовых воздействий напрямую в каждый регистр системы. Делается это при помощи так называемых сканирующих цепочек регистров. Ячейки сканирования объединяются в цепи, образуя сдвиговые регистры, которые соединяются со специализированным контроллером. Из тестового контроллера по такой цепи можно доставить нужные данные в любой регистр системы и считать в контроллер данные из любого регистра системы.

Тестовые цепи для аппаратной проверки микросхем, интерфейс и тестовый контроллер стандартизованы. Это стандарт IEEE 1149 (JTAG). Кроме устройств сканирования внутренних блоков, создается сдвиговый регистр, подключаемый к внешним выводам микросхемы. Такая цепь, называемая цепью периферийного сканирования (boundary scan chain), обеспечивает возможность подачи нужного воздействия на любой вывод микросхемы через тестовый контроллер и считывание логических состояний на всех выводах (рис. 8). В любой момент работа микросхемы может быть прервана и произведен доступ к определенным блокам по тестовому порту. Таким образом, этот порт может использоваться не только для тестирования микросхемы, но и для дальнейшего управления устройством. Например, через тестовый порт может загружаться встроенное ПО или подключаться внутрисхемный эмулятор – аппаратный отладчик ПО.

Проектирование блоков, осуществляющих тестирование, и генерация тестов – задача сложная, но формализуемая. Она с успехом может выполняться автоматически. Пакеты САПР и Cadence, и Synopsys содержат программы, обеспечивающие оптимальное подключение тестовых цепей и генерацию тестов аппаратного контроля. Важно отметить, что все тестовые схемы, за исключением сканирующих цепей, проектируются и включаются в систему еще на уровне разработки RTL кода.

Тестирование аналоговых и аналого-цифровых блоков СнК осуществляется отдельно. Обычно их интерфейсы полностью выводятся на внешние выводы микросхемы, позволяя получить всю необходимую информацию о функционировании этих блоков.

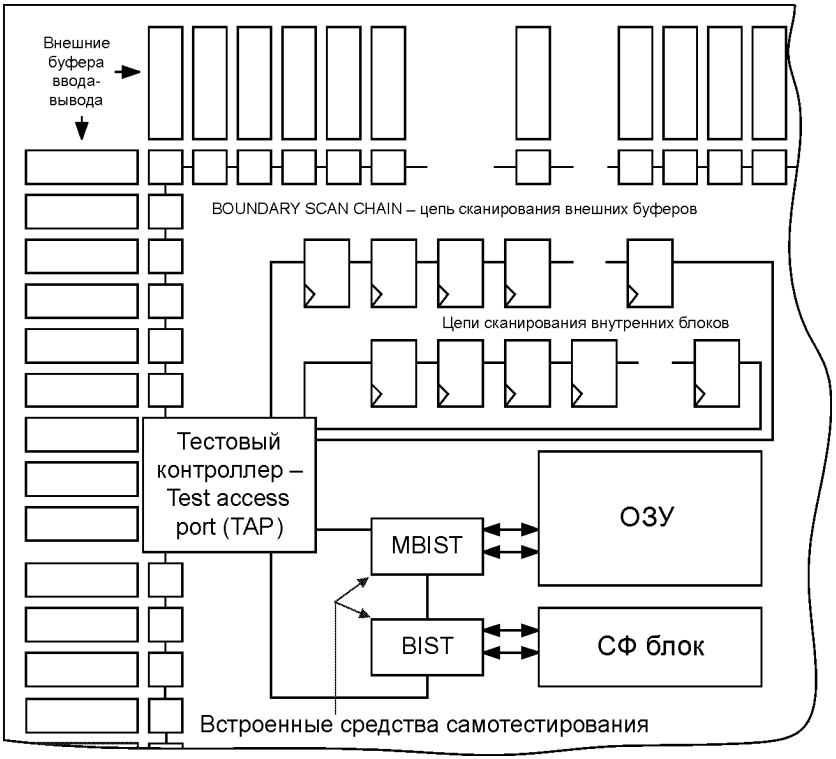


Рис. 8. Средства тестирования и самотестирования в СнК

Запоминающие устройства нерационально проверять как логические схемы, так как такая проверка потребует слишком большого набора тестовых векторов. Для блоков памяти в несколько мегабит число тестовых векторов может достигать десятков миллионов. Такие же проблемы возникают при тестировании некоторых сложнофункциональных блоков. В этих случаях более рационально применить встроенные аппаратные средства самотестирования – BIST, MBIST (Memory Build In Self Test). Пример их использования показан на рис. 8.

2. Методология проектирования аналого-цифровых схем

2.1. Введение

Процесс разработки аналого-цифровых схем существенно отличается от разработки полностью цифровых устройств. Проектирование цифровых схем ведется на системном (архитектурном) уровне и на уровне функциональных узлов с использованием СФ блоков. Разработка и оптимизация стандартных ячеек и блоков в большинстве случаев не имеет смысла и не делается, поскольку существует большой выбор готовых проверенных заготовок, поставляемых вместе с проектными библиотеками. Проектирование на вентильном и транзисторном уровне осуществляется лишь разработчиками библиотек элементов и блоков, а также при создании ИМС для специальных применений.

В цифровых проектах на первый план выдвигаются проблемы, связанные с большими размерами схем, когда число вентилей достигает десятков тысяч и более. Поэтому средства автоматизированного проектирования, включая средства синтеза схем и топологии из моделей уровня регистровых пересылок или моделей более высокого уровня, в таких проектах играют ключевую роль.

Задачи, возникающие при разработке аналого-цифровых схем, намного меньше обусловлены размерами проекта. Скорее они связаны со сложностями в достижении требуемых характеристик. Аналого-цифровые схемы состоят из уникальных блоков, разработку которых невозможно полностью автоматизировать. Их проектирование ведется вручную с учетом заложенных характеристик, а в случае чрезвычайно высоких требований помочь может только индивидуальный творческий подход разработчика. А творчество, как известно, не поддается автоматизации.

Сегодня такие компании, как Cadence, Synopsys, Magma, пытаются создать средства автоматизации синтеза аналого-цифровых схем, но большого успеха пока не достигнуто. Современные САПР позволяют синтезировать лишь относительно простые аналоговые схемы, имеющие средний уровень качественных показателей [8]. Очевидно, что стремительно растущая сложность аналого-цифровых систем, в конечном счете, потребует создания более раз-

витых средств автоматизации, но в ближайшей перспективе задачей САПР будет не замена разработчика, а эффективная помощь ему. Вместе с тем применение даже усовершенствованных средств разработки оказывается малоэффективным, если методология проектирования выбрана неверно. В 1998 г. компанией Collett International было проведено исследование производительности разработчиков микросхем. Этим показателем характеризовали нормализованное число транзисторов в спроектированных ИМС на одного человека в неделю. В исследовании анализировался процесс создания 20-ти микросхем специалистами 14-ти крупнейших микроэлектронных компаний. Оказалось, что разница в производительности труда разработчиков, оказавшихся на первом и последнем месте, была четырнадцатикратной. При этом дополнительная нагрузка и интенсификация труда работников не привела к существенному улучшению данного показателя. На основании анализа результатов исследования был сделан вывод, что главной причиной низкой производительности явилась неправильная методология проектирования. Компании, оказавшиеся на последних местах в рейтинге эффективности, использовали методологию проектирования «снизу вверх», а компании-лидеры использовали способ проектирования «сверху вниз». Остановимся подробнее на этих методах.

2.2. Метод проектирования «снизу вверх»

Это традиционный подход, когда разработка начинается с отдельных блоков, которые затем объединяются в систему. Проектирование блока начинается с формулирования требований и заканчивается его исполнением на уровне транзисторов. Каждый блок проверяется (верифицируется) как самостоятельная единица, а не как составная часть системы. В случае удовлетворения требованиям спецификации блоки собираются в элементы более высокого уровня, и процесс верификации повторяется. В конечном счете, достигается самый верхний уровень иерархии, и вся система оказывается собранной на уровне транзисторов. Такой метод, остающийся эффективным для небольших схем, имеет множество недостатков, проявляющих себя с ростом сложности проекта. Вот некоторые из них.

- В сложных проектах очень важно проводить моделирование на архитектурном уровне, потому что именно здесь закладываются функциональность и основные характеристики будущей системы. При разработке «снизу вверх» затруднительно выявить возможные структурные улучшения, понять, какие параметры и как влияют на систему, нельзя точно сформулировать требования к составным блокам.
- Когда блоки объединяются в систему, моделирование занимает значительное время, поэтому ее верификация затруднительна или вовсе невозможна.
- Любые ошибки, найденные на системном уровне, сложно исправить, так как потребуется переделывать блоки.
- Многие важные и трудоемкие этапы при использовании метода проектирования «снизу вверх» должны выполняться последовательно один за другим, поэтому сроки проекта затягиваются.

Первый из отмеченных недостатков можно исключить, не меняя в целом методологии проектирования. Строится модель архитектуры кристалла в виде блок-схемы, которая моделируется, например, в среде Matlab-Simulink или с использованием иного пакета высокоуровневой симуляции. Проводится полное исследование системы, определяются требования к отдельным блокам. А затем процесс разработки каждого блока ведется отдельно. Таким образом, процесс проектирования ведется по методу «снизу вверх», но ему предшествует архитектурное исследование. Недостатки, связанные с верификацией и временем на разработку, по-прежнему сохраняются.

2.3. Метод проектирования «сверху вниз»

Этот метод основан на последовательном систематическом переходе с архитектурного уровня на транзисторный. Каждый уровень тщательно верифицируется по формальному плану и только после этого осуществляется переход на следующий уровень детализации. Формализация процесса и взаимоотношений между разработчиками обеспечивает возможность эффективной работы над проектом параллельно нескольким группам разработчиков и даже удаленным рабочим группам. Необходимо соблюдать следующие принципы верификации.

- На любом этапе проекта архитектура остается открытой для всех разработчиков, каждый из которых моделирует свой блок в составе всей системы. При этом блоки могут быть описаны на поведенческом или на транзисторном уровне, или на уровне топологии.
- На протяжении всего процесса каждое внесенное изменение верифицируется в отлаженном на предыдущем шаге окружении.
- Вначале разрабатывается полный план верификации, которому придерживаются на всех стадиях проекта. План обязательно включает системную верификацию и смешанное моделирование. Он должен обеспечить максимальную эффективность и скорость симуляции, поэтому в сложных СнК моделирование всего кристалла с наибольшим уровнем детализации проводится редко.
- Разработка начинается с верхнего, поведенческого уровня и движется в направлении повышения степени детализации.

Важно правильно распределить роли в рабочей группе. Для успешного проекта необходим минимум один системный инженер, являющийся архитектором кристалла. Он должен быть достаточно опытным, обладать знаниями о системе в целом и о каждом из ее составных блоков. Именно он моделирует архитектуру, разрабатывает поведенческие модели всех составных блоков. Системный архитектор составляет верификационные планы, пишет спецификации на блоки и контролирует их разработку. Инженеры, проектирующие блоки, избавляются от этой работы.

2.4. Моделирование аналого-цифровых систем

Известно, что проектирование цифровых узлов хорошо автоматизировано и проходит по своему собственному циклу с применением специальных инструментов. В свою очередь, проектирование аналоговых блоков ведется вручную в своем цикле и другими средствами САПР. Оба цикла объединяются только на поздних этапах проектирования, когда собирается схема всего кристалла (рис. 9). В таком случае моделирование системы оказывается возможным только на транзисторном уровне, а ведение проекта в рамках методологии «сверху вниз» требует постоянного моделирования системы на всех уровнях абстракции.

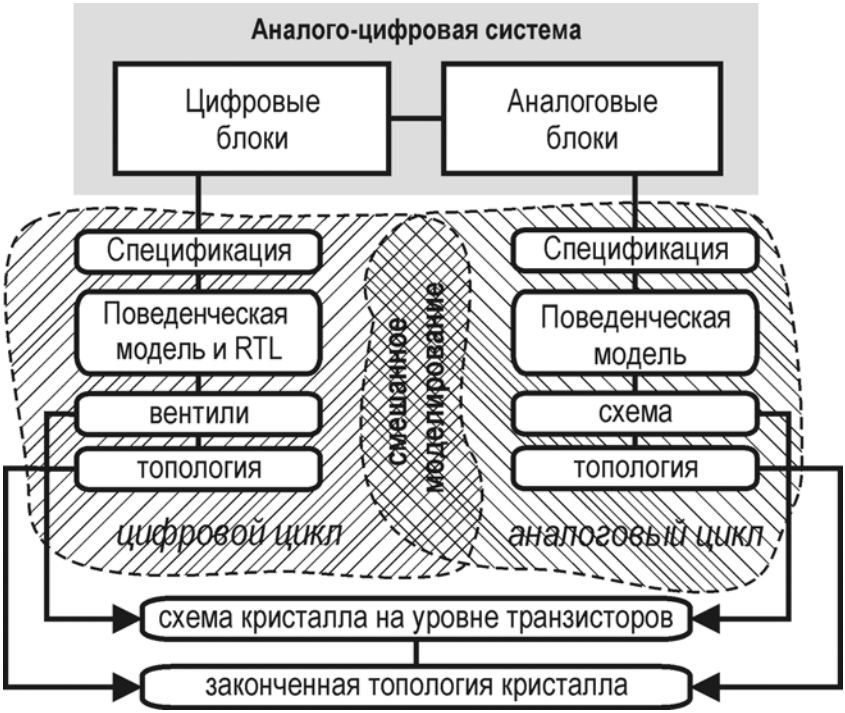


Рис. 9. Циклы моделирования в маршруте проектирования аналого-цифровых систем

Помочь в этой ситуации может так называемое смешанное моделирование, т.е. совместное моделирование цифровых и аналоговых блоков в составе системы. С помощью специальных симуляторов возможно совместное моделирование блоков любого уровня абстракции, разрабатываемых в разных циклах. Это позволяет непрерывно проверять функциональность аналоговых и цифровых блоков в реальном окружении, исследовать влияние одних блоков системы на другие и т. п. При использовании смешанного моделирования любой блок, разрабатываемый в цифровом цикле, может быть верифицирован в аналоговом окружении (рис. 10). Возможна и обратная ситуация. Если раньше можно было верифицировать аналого-цифровой блок, только когда цифровая часть синтезирова-

на до уровня транзисторов, то благодаря смешанному моделированию цифровые блоки можно моделировать на уровне исходного Verilog-описания. Это существенно снижает трудозатраты и время моделирования. В свою очередь, смешанный симулятор должен быть способен моделировать схему, часть компонентов которой находится на уровне Verilog, а другая часть – на транзисторном уровне.

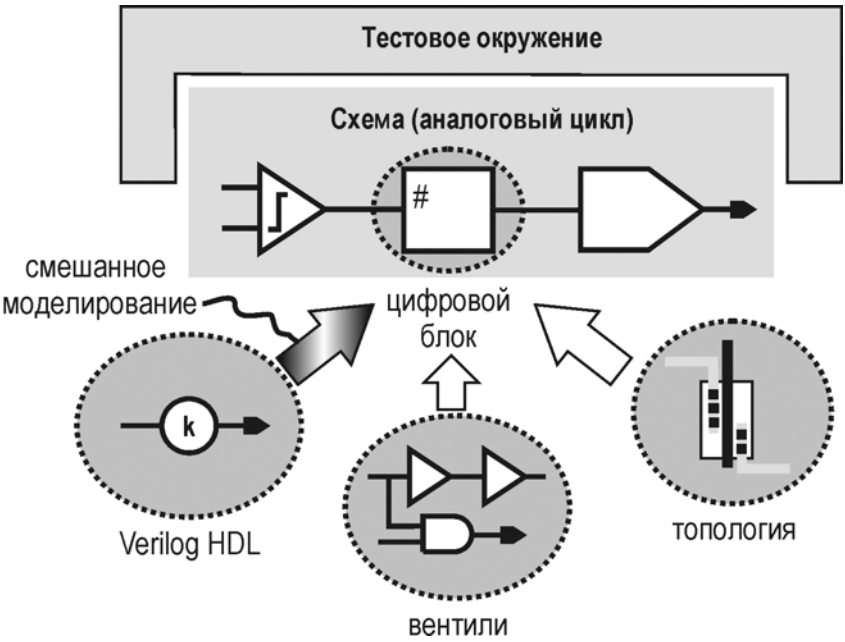


Рис. 10. Верификация цифровых блоков в аналоговом тестовом окружении

2.5. Цикл разработки аналого-цифровых ИМС средствами САПР Cadence

В следующих разделах будет рассмотрен пример проектирования аналого-цифровой схемы с использованием САПР компании Cadence Design Systems. Поэтому целесообразно показать, какими

программными средствами этой САПР реализуется методология проектирования «сверху вниз». На рис. 11 представлен маршрут проектирования ИМС.

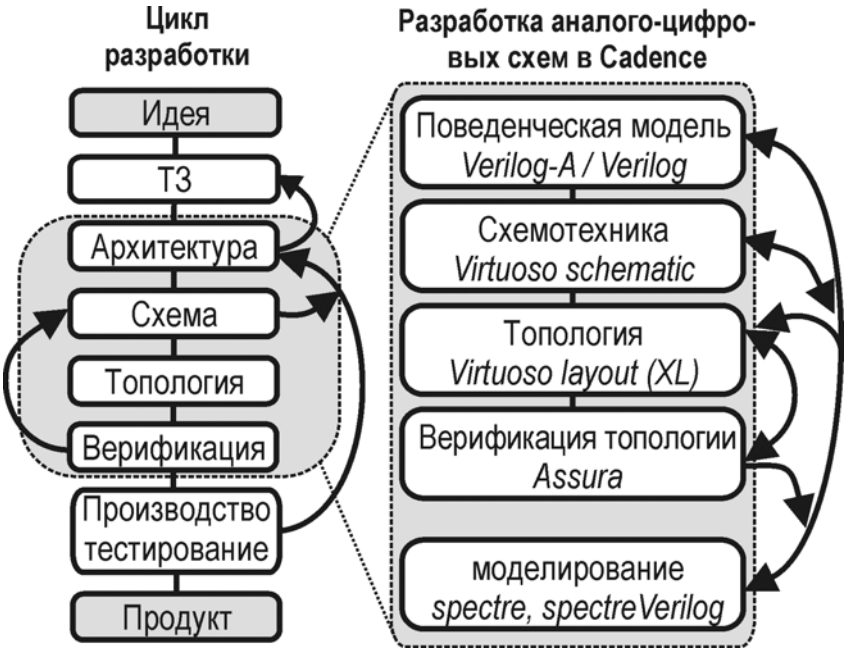


Рис. 11. Этапы маршрута проектирования ИМС, обеспечиваемые средствами САПР компании Cadence

На основе концепции или идеи формируется техническое задание (ТЗ) на продукт, учитывающее требуемые характеристики и условия эксплуатации. На основе данной спецификации разрабатывается и верифицируется архитектура. Затем итерационно проект проходит стадии разработки и верификации схемы и топологии. После окончательной верификации топология кристалла передается на фабрику. Изготовленные образцы проходят тестирование, где измеряются все параметры, заявленные в спецификации. Если по каким-либо критериям образцы не удовлетворяют требованиям ТЗ, весь процесс повторяется. Понятно, что проход каждой итера-

ции стоит дорого и занимает много времени, поэтому каждый выпуск микросхемы необходимо тщательно подготовить.

Часть цикла, реализуемая средствами САПР Cadence, показана в правой части рисунка. Эти стадии маршрута подробно рассмотрим на примере разработки реальной микросхемы.

Выводы

- Наибольшая производительность достигается, если проектирование ведется «сверху вниз».
- Сначала разрабатывается верхний, архитектурный уровень системы. При этом желательно сразу правильно разбить ее на блоки и определиться со всеми внутренними связями и портами блоков (pin-accurate system description).
- Когда система окончательно разбита на аналоговые и цифровые блоки, их совместное моделирование осуществляется с помощью смешанных симуляторов.
- Переходить на следующий уровень абстракции можно только после полной верификации предыдущего.
- Любое изменение верифицируется в уже проверенном окружении.

3. Пример проектирования аналого-цифрового блока СнК

3.1. Введение

В этом разделе на практическом примере рассмотрим полный цикл разработки микросхемы в САПР Cadence: от ТЗ до готового к отправке на фабрику материала. Основываясь на принципе разработки «сверху вниз», рассмотрим основные стадии проектирования, в том числе написание и тестирование моделей блоков (аналоговых и цифровых), создание схемы и топологии, формирование выходных баз данных.

Основной акцент будет сделан на ознакомлении с современными САПР, на освоении методов проектирования и получении практических навыков работы с типовыми программными средствами, а не на изучении схемотехники ИМС или конструкции полупроводниковых приборов. С другой стороны, не будем очень подробно рассматривать отдельные программные продукты, так как для них существуют хорошие описания с обучающими примерами.

Нашей целью не будет создание уникального или сложного прибора. Наоборот, постараемся дать подробное (шаг за шагом) описание процесса создания очень простого устройства, при разработке которого попробуем осветить как можно больше специфических моментов, связанных с проектированием аналого-цифровых схем.

Для этой цели в качестве хорошего примера для описания маршрута проектирования можно рассмотреть процесс разработки простейшего аналого-цифрового преобразователя (АЦП) параллельного типа. Подробное описание алгоритма работы таких преобразователей можно легко найти в учебной и специальной литературе [9].

Коротко остановимся на существенных для нас моментах. Вход V_{in} блока АЦП (рис. 12, А) служит для подачи преобразуемого аналогового сигнала. Вход синхронизации CLK служит для подачи тактового сигнала – последовательности импульсов прямоугольной формы. Поток цифровых отсчетов, который является результатом аналого-цифрового преобразования входного сигнала, подается на выход D_OUT. На вход V_{ref} подается опорное (эталонное) напря-

жение, которое определяет диапазон входных сигналов, т.е. размах полной шкалы преобразования.

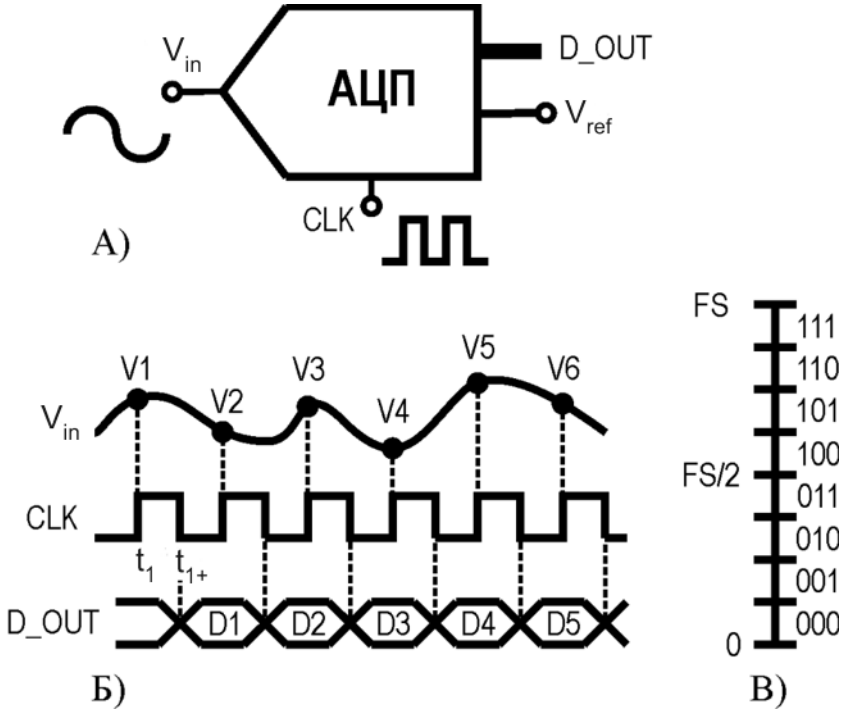


Рис. 12. Обозначение и порты подключения блока АЦП с однопроводным входом (А), временные диаграммы работы схемы (Б) и шкала преобразования (В)

Рассмотрим работу преобразователя с помощью временных диаграмм, показанных на рис. 12, Б. На вход V_{in} поступает переменное напряжение произвольной формы. Передние фронты тактовых импульсов на входе CLK определяют моменты выборки, т.е. те моменты времени, во время которых входное напряжение фиксируется – запоминается его мгновенное значение для последующего преобразования в цифровую форму. Предположим, что в момент времени t_1 величина входного напряжения оказалась равной V1. Тогда через промежуток времени ($t_{1+} - t_1$) по заднему фронту так-

товых импульсов на выходе D_OUT появится его цифровой эквивалент D1. Далее процесс дискретизации (взятия выборок) и квантования (аппроксимация амплитуды выборки цифровым кодом) повторяется, как показано на рис. 12, Б.

Цифровой код ставится в соответствие аналоговому напряжению с помощью шкалы квантования, показанной на рис. 12, В. Напряжение, соответствующее максимальному значению или пределу шкалы, принято обозначать FS (от Full Scale – полная шкала). В рассматриваемом примере разрядность АЦП выбрана равной трем битам. Поэтому полная шкала напряжений разбивается на восемь равных участков от 0 до FS, каждому из которых присваивается уникальный трехбитный код. Значение этого кода считается цифровым эквивалентом амплитуды любой выборки, если соответствующий ей уровень напряжения попал в границы данного интервала. Например, любые значения амплитуд выборок входного сигнала, лежащие в интервале от FS/2 до 5FS/8, преобразуются в код 100.

Отметим еще один аспект, касающийся архитектурного построения АЦП. Часто такие устройства имеют не одиночный вход, а два входа – неинвертирующий и инвертирующий (обозначены inp и inn на рис. 13, А), что позволяет осуществлять преобразование дифференциальных составляющих входных сигналов. В отличие от однопроводной схемы подачи сигнала на вход АЦП, показанной на рис. 12, А, когда уровень преобразуемого напряжения отсчитывается от уровня общей шины (земли), в двухпроводных схемах уровни сигналов представлены в дифференциальном виде как разность двух напряжений:

$$V_{in} = V_{inp} - V_{inn} .$$

Здесь V_{in} – дифференциальный входной сигнал, V_{inp} – напряжение относительно земли на не инвертирующем входе, V_{inn} – напряжение относительно земли на инвертирующем входе. Заметим, что дифференциальный входной сигнал может быть биполярным даже при одинаковой полярности сигналов на обоих входах АЦП, а его максимальный размах вдвое превышает допустимый диапазон входного сигнала недифференциальной схемы.

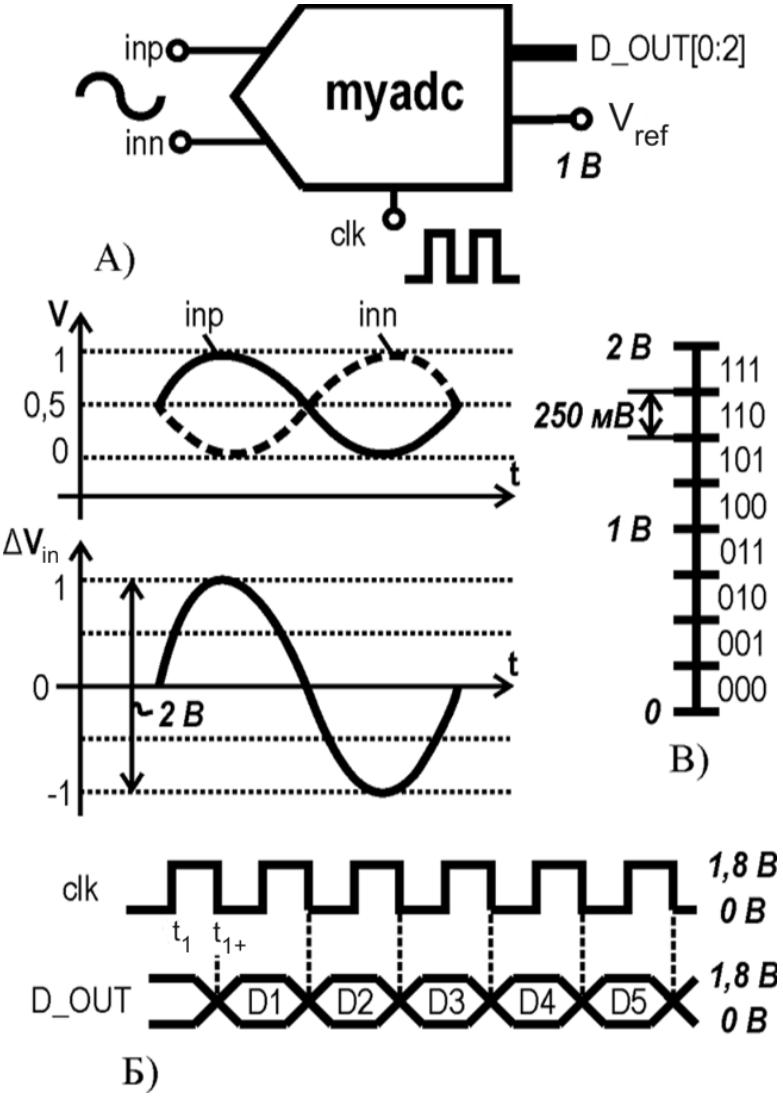


Рис. 13. Обозначение и порты подключения блока АЦП с дифференциальным входом (А), временные диаграммы работы схемы (Б) и шкала преобразования (В)

Использование дифференциальной архитектуры позволяет в большинстве случаев обеспечить также подавление синфазных составляющих обрабатываемых сигналов, что позволяет существенно снизить влияние помех и шумов, увеличить допустимый диапазон синфазных сигналов и имеет ряд других преимуществ. Воспользуемся этим подходом и сделаем нашу схему дифференциальной. Благодаря этому нам удастся увеличить допустимый входной размах и напряжение полной шкалы до 2 В при напряжении питания схемы всего 1,8 В.

3.2. Формулировка технического задания

Требуется разработать аналого-цифровой преобразователь параллельного типа, номинальной разрядностью 3 бита, который должен иметь следующие динамические параметры при частоте выборки 100 МГц и частоте входного сигнала 10 МГц: отношение сигнал–шум плюс гармонические искажения не менее 16 дБ, динамический диапазон свободный от гармоник не менее 22 дБ, эффективная разрядность 2,4 бита. Рабочий температурный диапазон от 0 до 70 ºС. Спецификация приведена в табл. 3.

Таблица 3

Параметр АЦП	Обозначения	Единицы измерения	Значения
Номинальная разрядность	N	бит	3
Частота преобразования	f_s	МГц	100
Отношение сигнал–шум плюс искажения	SNDR	дБ	16
Эффективная разрядность	ENOB	дБ	2,4
Динамический диапазон, свободный от гармоник	SFDR	дБ	22
Напряжение питания	V_{dd}	В	1,8
Уровень синфазного сигнала	V_{cm}	В	0,5
Диапазон входных сигналов	V_{in}	В	0 – 1,0
Дифференциальное напряжение полной шкалы	FS	В	2,0
Рабочий температурный диапазон	T	ºС	0 – 70

При ознакомлении с этим разделом предполагается знание принципов работы и основных параметров АЦП, что является предметом изучения в группе учебных дисциплин, посвященных измерительным системам и микросхемотехнике. Поэтому дадим лишь краткие определения некоторых из специфицируемых параметров.

- Разрядность формально определяется как двоичный логарифм максимального числа кодовых комбинаций на выходе, а фактически номинальная разрядность задается параметрами цифрового интерфейса АЦП.
- Частота преобразования – частота, с которой входной сигнал с заданными характеристиками преобразуется в выходной код при обеспечении требуемых значений заданных в спецификации параметров.
- Отношение сигнал–шум плюс гармонические искажения (динамический диапазон) можно определить как отношение среднего квадратического значения первой гармоники к среднему квадратическому значению шумов и высших гармоник в спектре выходного сигнала при подаче на вход синусоиды, двойная амплитуда которой устанавливается близкой к размаху полной шкалы АЦП.
- Эффективная разрядность – число разрядов, достигаемое при заданной частоте и амплитуде входного сигнала и при заданной частоте преобразования. Этот параметр определяется величиной реального динамического диапазона в соответствии с известным соотношением

$$ENOB = (SNDR - 1{,}76) / 6{,}02.$$

- Динамический диапазон, свободный от гармоник, – отношение среднего квадратического значения первой гармоники к среднему квадратическому значению наибольшей гармоники в спектре выходного сигнала при подаче на вход синусоиды, двойная амплитуда которой устанавливается близкой к размаху полной шкалы АЦП.

3.3. Создание модели АЦП на языке Verilog-A

Язык Verilog-A изначально был разработан как расширение Spice для высокого и низкого уровней абстракции. По аналогии с Verilog HDL язык Verilog-A эффективен как для описания высокоуровневых поведенческих моделей, так и низкоуровневых схем. Синтаксис языка практически полностью заимствован у Verilog HDL, а семантика соответствует стандартам Spice. Так, в языке поддерживаются принятые в Spice типы моделирования: временной, частотный и др. Имеется достаточно много литературы по Verilog-A с детальным описанием языка [10, 11], которую можно использовать для его самостоятельного изучения.

Язык Verilog-A позволяет описывать аналоговые или аналого-цифровые схемы с различным уровнем детализации. Поведенческое описание блоков системы дает возможность на ранних стадиях проекта проверять архитектурные решения без привязки к определенной технологии и без существенных временных затрат на детальную разработку и моделирование, а на поздних этапах проекта – оценивать влияние определенных низкоуровневых блоков на всю систему. Можно также существенно сократить время моделирования сложных систем путем замены части реальных блоков их поведенческим описанием. Кроме того, с помощью Verilog-A можно:

- управлять сложностью проекта при разработке сверху вниз;
- писать компактные модели независимо от реального исполнения блока;
- использовать одни и те же блоки в разных проектах независимо от технологии;
- передавать или продавать модели СФ блоков, поскольку язык Verilog-A является общепринятым стандартом.

Скажем еще несколько слов о языке Verilog-AMS. Основная задача языка – компактное описание аналого-цифровых систем. В нем совмещены возможности языков Verilog-A и Verilog HDL, что ускоряет процесс разработки модели. Кроме того, при моделировании блоков на Verilog-AMS привлекается только один симулятор в отличие от смешанных проектов на Verilog-A и Verilog, для расчета которых нужно два симулятора.

Основной недостаток Verilog-AMS заключается в том, что он не может быть использован на всех этапах ведения проекта с приме-

нением методологии проектирования «сверху вниз». Написав модель сложного блока на Verilog-AMS, у нас не будет возможности двигаться дальше, понижая уровень абстракции.

Это связано со спецификой САПР, так как инструменты для разработки аналоговых и цифровых блоков – разные, и все равно придется делить систему на блоки Verilog-A и Verilog HDL.

3.3.1. Алгоритм работы АЦП

В соответствие с предложенной методикой начнем разработку устройства с создания поведенческой модели. На языке Verilog-A опишем алгоритм работы АЦП. Сразу оговоримся, что на данном этапе проекта нас не интересует, какова будет в дальнейшем реальная реализация устройства. Модель описывает лишь поведение блока, представляемого в виде некоторого «черного ящика». Значит, что текст модели можно оптимизировать, добиваясь читаемости кода, простоты и, главное, максимальной скорости симуляции.

Самый простой способ найти соответствие между выходным кодом и аналоговым сигналом – разбить всю шкалу на восемь равных участков и поочередно проверять, принадлежит ли какому-нибудь из них значение амплитуды выборки. Для трех бит нужно описать восемь операций сравнения. А если бы пришлось писать модель 8-битного АЦП, то операций потребовалось бы уже шестьдесят четыре. Мы оптимизировали алгоритм, описав только три операции сравнения. Данный алгоритм легко расширяется для любой разрядности преобразователя с помощью оператора цикла, и в этом случае число сравнений равно числу бит.

Алгоритм поясним на примере (рис. 14). Пусть в момент выборки фиксируется напряжение V_sample, лежащее в диапазоне [-1; 1]. Предварительно избавимся от отрицательных чисел, сдвинув значение V_sample на половину шкалы вверх. Теперь V_sample лежит в диапазоне от 0 до FS, где $FS = 2 \cdot V_{ref}$ это напряжение полной шкалы. Далее последовательно находим все биты выходного кода, начиная со старшего разряда.

Старший бит принимаем равным логической "1", если V_sample лежит в верхней половине указанного диапазона, или принимаем равным логическому "0" если V_sample лежит в нижней половине диапазона. В случае если старший бит равен логической "1" опус-

каем V_sample на полшкалы вниз, сужая, таким образом, диапазон. Новые границы диапазона – 0 и FS/2.

Второй и третий биты находим аналогично, рассматривая диапазоны, соответственно, от 0 до FS/2 и от 0 до FS/4. Если получено нулевое значение бита, то сдвига не требуется, так как обрабатываемая выборка уже находится в нужном диапазоне.

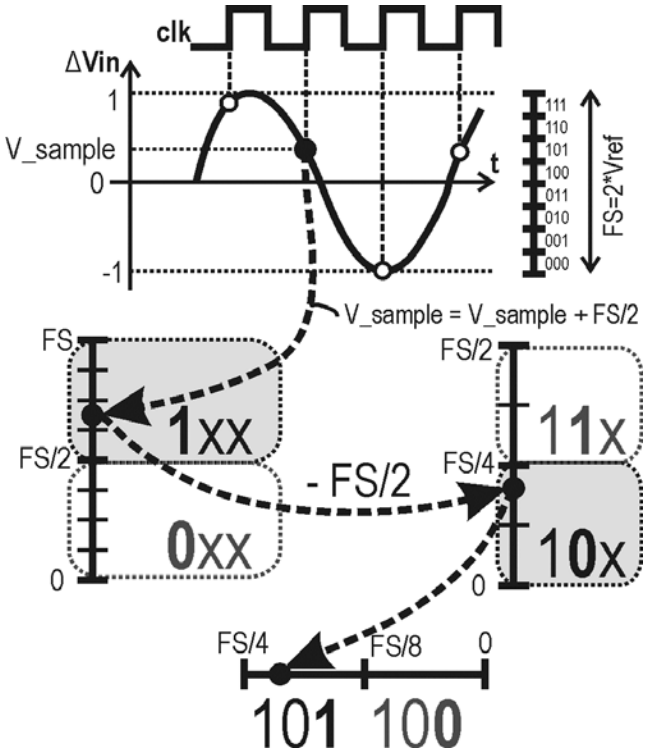


Рис. 14. Иллюстрация к моделированию алгоритма работы АЦП

Текст модели приведен далее в листинге 1. Остановимся на нем подробнее. После объявления портов декларируем параметры модели. К ним относятся времена фронтов и задержек по выходной цифровой шине, а также значения напряжений, соответствующие логическим уровням (рис. 15).

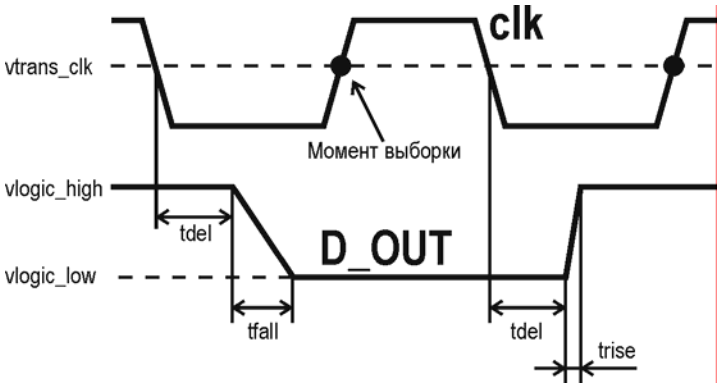


Рис. 15. Временные параметры и логические уровни в модели АЦП

Уровень входного тактирующего сигнала, по которому определяется момент выборки, задан параметрически. Момент выборки фиксируется по переднему фронту сигнала clk стандартным оператором **cross** с параметром +1. В этот момент во внутренние переменные запоминается входное дифференциальное напряжение V_sample (сразу со сдвигом) и размер шкалы FS. По заднему фронту сигнала clk изменяются значения выходных бит в соответствии с алгоритмом. Все операции производятся с помощью блокирующего оператора " = " над внутренними переменными, которые параллельным оператором " <+ " и функцией **transition** ставятся в соответствие выходным напряжениям шины D_OUT.

Листинг 1

```
// Verilog-A модель АЦП
// Частота преобразования 100 МГц
// Разрядность 3 бита
`include "constants.h"
`include "disciplines.h"
module myadc(D_OUT, Vref, clk, inn, inp);

// описание портов
output [0:2] D_OUT;   // выходная шина
```

```
electrical [0:2] D_OUT;
input Vref;              // опорное напряжение
electrical Vref;
input clk;               // тактовый сигнал
electrical clk;
input inn;               // инвертирующий аналоговый вход
electrical inn;
input inp;               // неинвертирующий аналоговый вход
electrical inp;

// описание параметров модели
// параметры шины D_OUT
parameter real trise = 1p from [0:inf); // время нарастания
parameter real tfall = 1p from [0:inf); // время спада
parameter real tdel = 0 from [0:inf);   // задержка перехода
parameter real vlogic_high = 1.8;       // уровень логической “1”
parameter real vlogic_low  = 0;         // уровень логического “0”

// порог переключения тактового сигнала
parameter real vtrans_clk    = 0.9;

// описание внутренних переменных
real V_sample;           // выборка
real vd[0:2];            // внутренний эквивалент D_OUT
real FS;                 // полная шкала

analog begin
        // момент выборки
        @ (cross(V(clk) - vtrans_clk, +1)) begin
                FS=2*V(Vref);
                V_sample=V(inp)-V(inn) + FS/2;
        end

        // формирование выходного кода
        @ (cross(V(clk) - vtrans_clk, -1)) begin

                //находим старший бит
                if (V_sample>FS/2) begin
```

```
                vd[0] = vlogic_high ;
                V_sample=V_sample-FS/2;

            end else
                    vd[0]=vlogic_low;

            //находим второй бит
            if (V_sample>FS/4) begin
                vd[1] = vlogic_high ;
                V_sample=V_sample-FS/4;

            end else
                    vd[1]=vlogic_low;

            //находим младший бит
            if (V_sample>FS/8) begin
                vd[2] = vlogic_high ;
                V_sample=V_sample-FS/8;

            end else
                    vd[2]=vlogic_low;

        end

        // ставим в соответствие портам
        V(D_OUT[0]) <+ transition(vd[0], tdel, trise, tfall);
        V(D_OUT[1]) <+ transition(vd[1], tdel, trise, tfall);
        V(D_OUT[2]) <+ transition(vd[2], tdel, trise, tfall);

end
endmodule
```

3.3.2. Средства проектирования компании Cadence

Здесь и далее поговорим о том, как реализовывать каждый из этапов проекта, используя инструментальные средства Cadence. Начинаем знакомиться с этой системой проектирования, которая в настоящее время фактически является мировым индустриальным

стандартом при создании микросхем всех типов и уровней интеграции.

САПР такой сложности чувствителен к множеству настроек, параметров окружения, и, конечно, к операционной системе, в которой он работает. Поэтому укажем, в какой среде будем работать и какое программное обеспечение использовать (табл. 4).

Таблица 4

Наименование пакетов программ	Версия, * подверсия
Операционная система	
Red Hat Enterprise Linux 4	ядро Linux 2.6.9-22.Elsmp EDT 2005
САПР Cadence Design System Inc.	
IC	icfb 5.1.0 05/07/2006, * 5.10.41.500.3.41 (32-бит)
Hierarchy Editor	05.01.000-s 41, 2004
Layout editor	5.1.0
IUS	5.6, 2005
Assura	5.1.41
Пакет проектных библиотек (Process Design Kit, PDK)	
XFAB XC018	2.1

Прежде чем открыть проект и начать работу, скажем несколько слов о том, как Cadence хранит нашу информацию. Все файлы проекта хранятся на диске в виде структурированной базы данных. При этом соблюдается следующая иерархия.

- *Библиотеки* (libraries) – самый верхний уровень базы данных. Любой проект состоит из одной или нескольких библиотек. Библиотеки могут иметь свой собственный технологический файл или быть подключенными к технологическому файлу другой библиотеки. Библиотека хранится в виде каталога с определенным набором служебных файлов [12].
- *Ячейки* (cells) входят в состав библиотек. Ячейка может представлять собой законченный блок (instance), который используется в других блоках, или же быть элементом верхнего уровня, например тестовым окружением или топологией микросхемы.

- *Виды* (views) – подуровни ячеек, самостоятельные единицы, которыми оперирует система, компонуя список цепей (нетлист, netlist). С помощью видов можно легко конфигурировать проект для расчета. Виды бывают многих типов, но для нас интерес представляют только приведенные в табл. 5.

Таблица 5

Тип вида	Обозначение по умолчанию	Описание
Конфигурация	config	Хранит конфигурацию проекта для моделирования
Схема	schematic	Схема блока, доступная для редактирования в Virtuoso Schematic Editor
Символ	symbol	Пиктограмма блока, входящего в состав схемы. Может редактироваться в Virtuoso Symbol Editor. Этот вид должен обязательно присутствовать, если блок используется в иерархии
Модель Verilog-A	veriloga	Код на языке Verilog-A, описывающий интерфейсы и поведение блока
Модель Verilog HDL	verilog functional	Код на языке Verilog HDL, описывающий интерфейсы и поведение цифрового блока
Топология	layout	Топология блока, доступная для редактирования в Virtuoso Layout Editor
Схема с паразитными элементами	av_extracted	Схема, полученная в результате экстракции паразитных элементов из топологии. Недоступна для редактирования

Управление базой данных: библиотеками, ячейками и видами – осуществляется с помощью окна Library Manager, подробную инструкцию для которого можно найти в [12].

3.3.3. Практическая работа в Cadence. Создание простой модели АЦП

Итак, начинаем реализацию первого этапа проекта.

- Запускаем Cadence IC, перейдя в консоли Linux в рабочую папку и выполнив команду icfb&.
- Из окна Library Manager (меню «Tools/Library Manager…» главного окна CIW) создаем рабочую библиотеку под названием pract_sample и прикрепляем ее к технологическому файлу нашего пакета проектных библиотек.
- Выделив новую библиотеку в окне Library Manager, создаем новую ячейку, воспользовавшись меню «File/New/Cell View…». В открывшемся окне выбираем «Tool – VerilogA-Editor». В поле «Cell Name» вводим myadc.
- В открывшееся окно редактора с текстом заготовки для нашего модуля вводим текст модели (листинг 1). Закрывая окно, Cadence проверит синтаксис и в случае удачи откомпилирует код. Если в коде будут найдены ошибки, они отразятся в высветившемся окне, и вам предложат их исправить. По умолчанию в Cadence используется традиционный Linux редактор VI. Если он кажется неудобным, то можно использовать более «дружелюбный» редактор, например NEdit. Для этого данный компонент должен быть установлен в операционной системе и должна быть определена переменная окружения *EDITOR = nedit* .
- Далее появится диалоговое окно с предложением создать вид символа (symbol) на основе описания интерфейсов в тексте модели. Согласимся, но после создания в открывшемся окне Virtuoso Symbol Editing отредактируем пиктограмму, как показано на рис. 16.

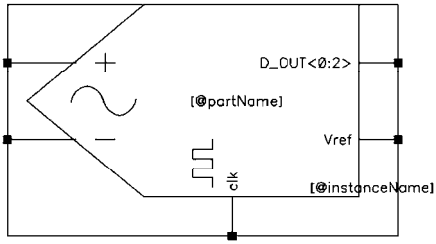


Рис. 16. Предлагаемый вариант изображения символа АЦП

Таким образом, в библиотеке pract_sample появилась ячейка myadc с двумя видами: veriloga и symbol. Данный блок будет верхним уровнем иерархии, и далее начнем его тестирование.

Выводы

Были сформулированы требования к АЦП и написана Verilog-A модель идеального устройства, также научились создавать виды veriloga и symbol для разрабатываемого устройства.

3.4. Тестирование модели АЦП

Для тестирования модели нашего АЦП предлагается схема, показанная на рис. 17. На входы inp и inn подадим дифференциальный синусоидальный сигнал, на вход clk – прямоугольные импульсы периодом 10 нс и скважностью 50 %, а на Vref – постоянное напряжение величиной 1 В.

Удобно сравнивать выходной и выходной сигналы тестируемого блока, если оба сигнала имеют одинаковую форму представления. Наиболее наглядно представление сигналов в аналоговом виде. Преобразовать выходной цифровой код АЦП в эквивалентное ему напряжение нам поможет идеальный ЦАП.

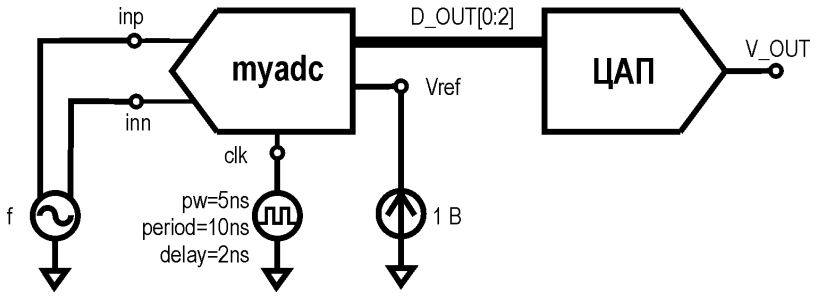


Рис. 17. Схема тестирования модели АЦП

Модель идеального ЦАП разрядностью 3 бита на языке Verilog-A приведена в листинге 2. Для простоты блок работает асинхронно, а выходное напряжение получается как результат двоичного взвешивания входных кодов.

Листинг 2

```
// Verilog-A модель идеального ЦАП
module dac_3bit(vd2, vd1, vd0, vout);

input vd2, vd1, vd0;     // цифровые входы

output vout;             // аналоговый выход
electrical vd2, vd1, vd0, vout;

parameter real vrefP  = 1;       // верхний предел шкалы
parameter real vrefN  = -1;      // нижний предел шкалы

parameter real trise = 1p from [0:inf);
parameter real tfall = 1p from [0:inf);
parameter real tdel  = 0 from [0:inf);
parameter real vtrans  = 0.9;    // порог переключения

   real out_scaled;
   real vout0;

   analog begin

                out_scaled = 0;

                out_scaled = out_scaled + ((V(vd0) > vtrans) ? 4 : 0);
                out_scaled = out_scaled + ((V(vd1) > vtrans) ? 2 : 0);
                out_scaled = out_scaled + ((V(vd2) > vtrans) ? 1 : 0);

                vout0 = vrefN+(vrefP-vrefN)*out_scaled/(pow(2,3)-1);

                V(vout) <+ transition( vout0, tdel, trise, tfall );

   end

endmodule
```

В результате тестирования АЦП нужно измерить его динамические параметры: отношение сигнал–шум плюс искажения (SNDR) и динамический диапазон, свободный от гармоник (SFDR). Эти параметры легко извлечь из спектрограммы, полученной преобразованием Фурье выходной последовательности кодов АЦП. С тем, как это делается в Cadence, будем знакомиться [13]. Заметим, что дальнейшее продвижение по маршруту проектирования подразумевает наличие базовых знаний по теории спектрального анализа непрерывных и дискретных сигналов. Минимальные сведения даются дальше.

3.4.1. Сведения по спектральному анализу сигналов

Прямое и обратное преобразования Фурье устанавливают связь между представлением сигналов во временной и частотной области. Для анализа дискретных последовательностей применяют дискретное преобразование Фурье (ДПФ). Вычисление ДПФ обычно осуществляют, используя эффективные алгоритмы быстрого преобразования Фурье (БПФ).

Описание преобразования Фурье можно найти в каждой книге по цифровой обработке сигналов [14]. Поэтому рассмотрим лишь основные соотношения, необходимые для дальнейшего обсуждения.

На рис. 18 дана иллюстрация представления сигнала во временной и частотной области. Переход от временного представления к частотному осуществляется с помощью БПФ по N выборкам сигнала с шагом дискретизации Δt. Параметр $F_S = 1/\Delta t$ называется частотой дискретизации БПФ. Результат преобразования представляет собой дискретную последовательность из N точек с шагом дискретизации Δf, называемым разрешением БПФ по частоте. Эта последовательность характеризует амплитуды спектральных компонент сигнала на частотах от 0 (постоянная составляющая) до $F_S/2 - \Delta f$.

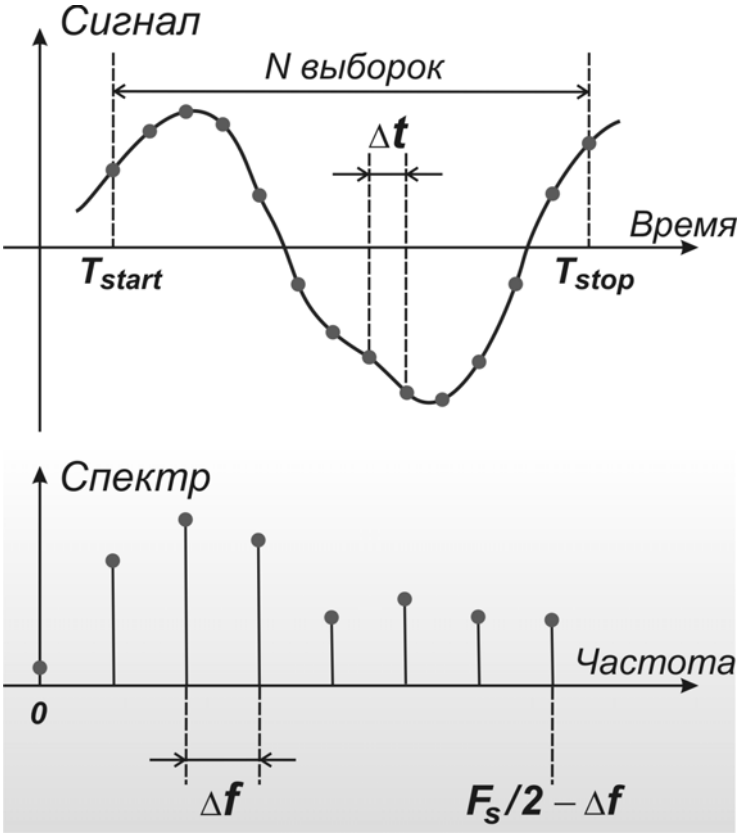


Рис. 18. Представление сигнала во временной и частотной областях

Основные свойства БПФ.

- Разрешение по частоте прямо пропорционально частоте дискретизации и обратно пропорционально числу точек БФП:

$$\Delta f = \frac{1}{N\Delta t} = \frac{F_S}{N}.$$

- Интерес представляет лишь первая половина выходных данных от 0 до $N/2 - 1$. Вторая половина последовательности от $N/2$ до $N - 1$ – это отражение первой относительно частоты $F_S/2$, не содержащая дополнительной информации.

- В отсутствие эффекта «утечки» спектра, о котором речь пойдет ниже, точность расчета спектральных компонент с помощью встроенных функций БПФ определяется только точностью вычисления прямого ДПФ:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j 2\pi k n / N}, \quad k = 0, 1, \ldots N-1,$$

где $X(k)$ – элемент выходной последовательности БПФ; $x(n)$ – элемент входной выборки из N отсчетов; $j = \sqrt{-1}$.

- Использование алгоритма БПФ возможно только, если число точек N является степенью двойки, иначе необходимо применять более медленный алгоритм ДПФ.

3.4.2. Реализация методов спектрального анализа в Cadence

САПР Cadence не имеет встроенной функции вычисления ДПФ. В распоряжении пользователя имеется лишь функция БПФ (dft) со следующими входными параметрами:

- результаты анализа переходных процессов во временной области и интервал моделирования, на котором берутся выборки;
- число точек преобразования;
- используемое окно и параметры окна.

Для получения достоверных результатов необходимо правильно задать все параметры функции. Рассмотрим этот вопрос подробнее.

Число точек БПФ и интервал моделирования. Для получения точных результатов, число выборок N должно быть степенью 2. Если это не так, то Cadence округляет выражение $\log_2(N)$ до ближайшего целого числа и проводит преобразование, используя это новое число выборок. Например, если есть 10 выборок, по которым требуется провести БПФ, то Cadence сначала округлит N до 16 и получит недостающие выборки путем интерполяции. Полученный результат БПФ будет искажен гармониками вследствие интерполяции. Таким образом, число точек преобразования определяется из следующего условия:

$$N = F_S / \Delta f = 2^k, \quad k = 1, 2, \ldots$$

В общем случае частота дискретизации БПФ F_S выбирается в соответствии с критерием Котельникова (Найквиста), чтобы избежать искажений вследствие эффекта наложения спектров. Однако, при моделировании таких устройств, как ЦАП и АЦП, этот параметр соответствует частоте, с которой эти устройства выполняют преобразование данных. Зная число точек N и частоту дискретизации F_S, можно задать временной интервал для функции БПФ:

$$T_{\text{stop}} - T_{\text{start}} = N / F_S ,$$

где T_{stop} – время окончания, а T_{start} – время начала интервала.

Интерполяция при вычислении БПФ. Рассмотрим пример получения в Cadence спектра синусоидального сигнала с частотой f_{in} = 312,5 кГц. Для этого проведем БПФ по N = 128 точкам с частотой дискретизации F_S = 10 МГц. В качестве входных данных возьмем результаты анализа во временной области на интервале от 0 до N/F_S = 12,8 мкс. Результат, показанный на рис. 19, может показаться странным. Во-первых, спектр содержит значительные гармонические компоненты, а во-вторых, уровень шума выше –80 дБ. Это соответствует соотношению сигнал/шум 50 – 60 дБ. Учитывая, что для задач спектрального анализа БПФ должно обеспечивать соотношение сигнал/шум не менее 120 дБ, полученный результат явно неудовлетворительный.

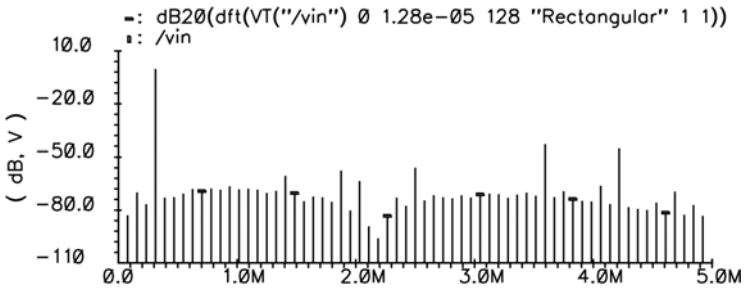


Рис. 19. Спектр синусоидального сигнала, полученный в Cadence

Известны причины погрешности вычисления БПФ: «утечка» спектра, наложение спектров, шум симулятора. Однако наибольшая ошибка возникает, если входные отчеты были получены с ис-

пользованием интерполяции. Так как симулятор Cadence при временном анализе генерирует точки с переменным шагом, а анализ Фурье оперирует с равномерно распределенными по времени отчетами, полученные данные интерполируются перед подачей на вход функции БПФ. На рис. 20 приведена разность между точным синусоидальным сигналом из нашего примера и сигналом, который использовался Cadence при вычислении БПФ. Ошибка вследствие линейной интерполяции достигает 0,2 %.

В результате спектр интерполированного сигнала (см. рис. 19) содержит сумму спектра идеальной синусоиды и спектра сигнала ошибки, что серьезно ограничивает разрешающую способность БПФ. Чтобы решить проблему, можно повысить требования к точности расчета и/или уменьшить шаг моделирования. Данный метод не всегда дает хорошие результаты, вместе с тем замедляет расчет и может привести к проблемам со сходимостью. Лучший способ – заставить симулятор рассчитать схему в каждой точке, используемой в качестве входного отчета БПФ. Для этого в Cadence есть специальные параметры, доступные в окне настройки временного анализа (рис. 21). Указав значение параметра *strobeperiod* можно задать симулятору период времени, через который нужно сохранять данные. Параметры *skipstart* и *strobedelay* отвечают за временной и фазовый сдвиги момента начала записи.

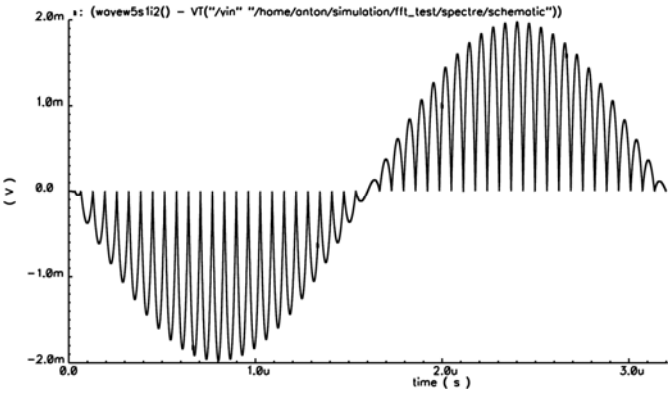


Рис. 20. Разница между точным и линейно-интерполированным синусоидальным сигналом

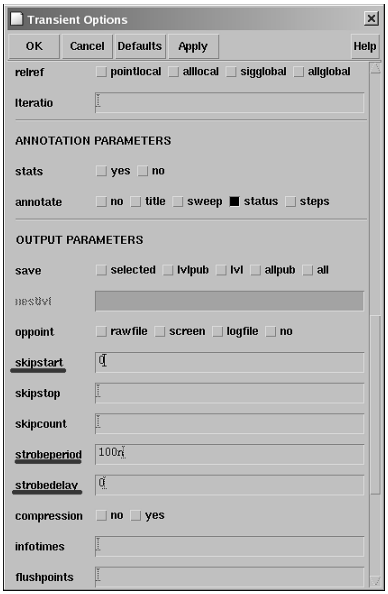


Рис. 21. Настройка параметров моделирования для БПФ

Зададим *strobeperiod* = 1 / F_S = 100 нс, *skipstart* и *strobedelay* равными 0, так как сохранять данные временного анализа будем с момента начала симуляции. Результаты моделирования приведены на рис. 22. Видно, что гармонические искажения исчезли, а уровень шума опустился до –300 дБ. Теперь этот уровень определяется только точностью вычислений симулятора.

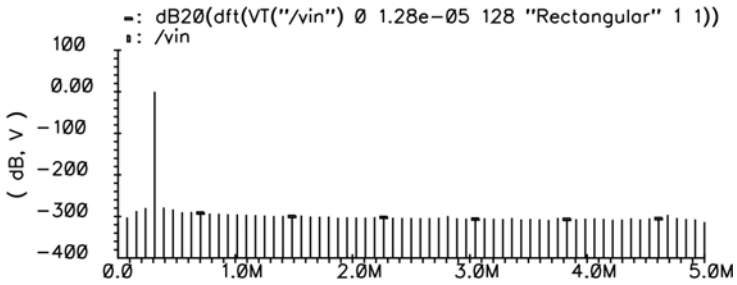


Рис. 22. Спектр синусоидального сигнала, полученный в Cadence с опцией strobeperiod

Эффект «утечки» спектра. Настроив опции симулятора, попробуем поменять в нашем примере частоту синусоидального входного сигнала с 312,5 на 300 кГц. Результирующий спектр показан на рис. 23. Он искажен, а увеличение числа точек БПФ не решает этой проблемы.

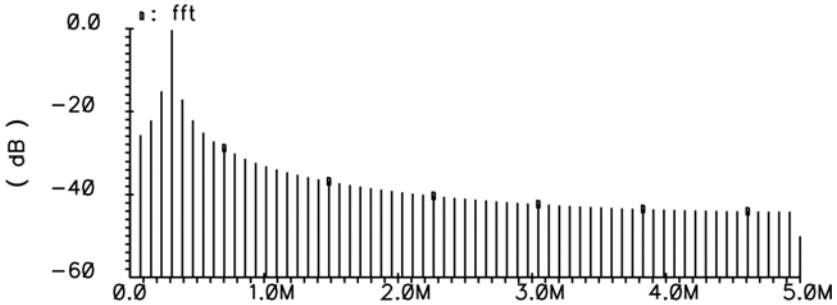


Рис. 23. Эффект «утечки» спектра вследствие проведения БПФ по нецелому числу периодов входного сигнала

В предыдущем примере входная частота 312,5 кГц была выбрана неслучайно. Для времени расчета 12,8 мкс она соответствует ровно четырем периодам синусоидального сигнала. Частота 300 кГц – это 3,84 периода. Спектр сигнала с нецелым числом периодов искажается из-за так называемого эффекта «утечки» спектра.

Этот эффект подробно рассмотрен в книгах по цифровой обработке сигналов и заключается в перераспределении энергии компонент спектра по всем выходным отчетам БПФ (бинам) вследствие особого вида собственной амплитудно-частотной характеристики БПФ. Сравнив рис. 22 и 23, можно увидеть, что энергия основного бина как бы «растекается» по всем остальным.

Спектральная «утечка» может быть ослаблена применением различного вида окон (рис. 24), которые корректируют частотную характеристику БПФ. Однако полностью влияние эффекта не устраняется, поэтому, если есть возможность, лучше проводить преобразование по целому числу периодов входного сигнала, т.е. частота входного сигнала должна быть равна:

$$f_{in} = m / (T_{start} - T_{stop}),\ m = 1, 2, 3, \ldots, \qquad (*)$$

где m – целое число периодов. Сравнение двух методов снижения влияния «утечки спектра» приведено в табл. 6.

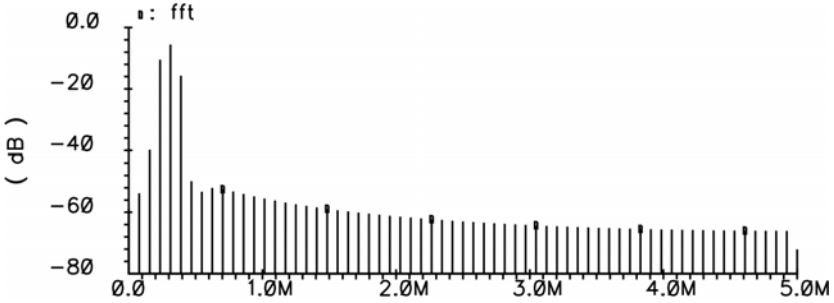


Рис. 24. «Утечка» спектра ослаблена из-за использования окна Хэмминга

Таблица 6

Задание длительности анализа равной целому числу периодов сигнала	Использование окна
Тестовый сигнал попадает точно в один бин.	Нет ограничений на частоту сигнала.
Требуется тщательный выбор частоты сигнала.	Требуется больше выборок для достижения заданной точности.
Хорошо подходит для моделирования	Расширение спектральных линий

Методику вычисления БПФ в Cadence можно условно изобразить в виде блок-схемы, показанной на рис. 25.

Отметим, что не рекомендуется выбирать частоту входного сигнала кратной частоте дискретизации F_S. В случае кратности этих частот входные отчеты будут периодически повторяться через каждые F_S/f_{in} выборок. Это может привести к тому, что равномерно распределенная по всей полосе энергия шума сконцентрируется в нескольких отдельных составляющих спектра вокруг основной частоты.

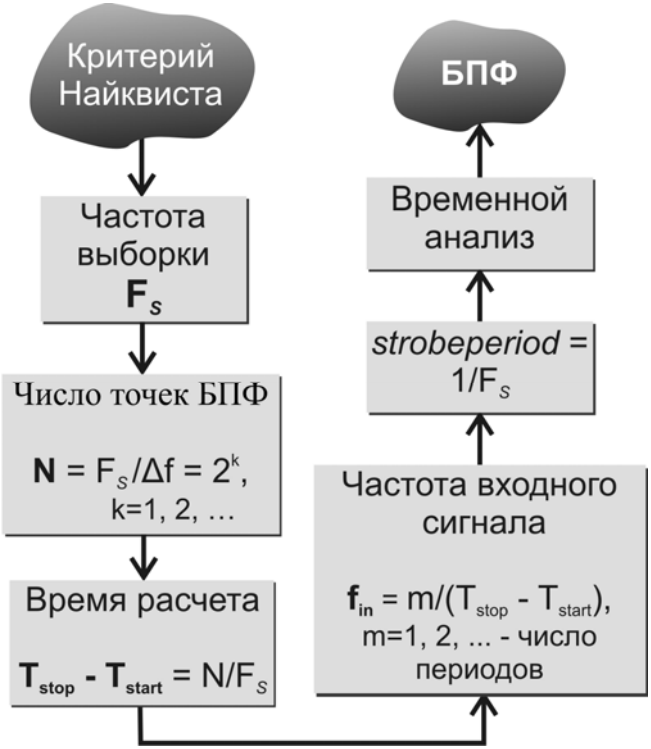


Рис. 25. Методика спектрального анализа в Cadence

Для определения параметров БПФ в нашем случае воспользуемся методикой спектрального анализа, показанной в виде структурной схемы на рис. 25. Частоту дискретизации F_S выберем равной частоте выборки АЦП – 100 МГц. Число точек БПФ – 4096, что соответствует разрешению по частоте Δf приблизительно 24,4 кГц. Время расчета $T_{stop} - T_{start}$ получается равным 40,96 мкс.

Следующий шаг – выбор частоты входного синусоидального сигнала. Допустим, нужно определить динамические параметры АЦП на частоте около 10 МГц. Задавать частоту f_{in} точно равной 10 МГц нельзя, так как, во-первых, входной сигнал не будет иметь целого числа периодов в 40,96 мкс, а во-вторых, эта частота кратна частоте дискретизации F_S. Воспользуемся формулой (*) и, переби-

рая целое число периодов m, найдем частоту, ближайшую к 10 МГц. Например, для m = 400 частота f_{in} оказывается приблизительно равной 9,77 МГц. Эту частоту и зададим входному источнику. Для повышения точности в параметрах источника сигнала Cadence лучше писать не округленное число, а выражение «1/102.4ns». Время расчета 41,09 мкс выбираем с запасом 130 нс на установление выходного сигнала.

3.4.3. Практическая работа в Cadence. Тестирование модели

Повторим последовательность действий, проделанных на этапе 1 для модели myadc, и создадим в библиотеке pract_sample блок dac_3bit. В нем также будет вид veriloga на основе модели из листинга 2 и вид symbol. Создадим схему тестирования нашего блока myadc. Для этого выполним следующие действия.

- В библиотеке pract_sample создадим новую ячейку типа schematic под названием test_bench.
- В открывшемся окне редактора принципиальных схем Virtuoso Schematic Editing [15] соберем схему, показанную на рис. 17. Нажав "**i**" расположим элементы myadc и dac_3bit из библиотеки pract_sample. В качестве источника постоянного напряжения Vref выберем элемент vdc из библиотеки analogLib. Источником тактового сигнала будет служить генератор vpulse из той же библиотеки со следующими параметрами: **Voltage 1** = 0, **Voltage 2** = 1.8, **Delay Time** = 2n, **Rise time** = 1p, **Fall time** = 1p, **Pulse width** = 5n, **Period** = 10n.
- В качестве источника входного дифференциального сигнала синусоидальной формы можно использовать два синус-генератора vsin из analogLib. Частоту зададим как «1/102.4n» (см. далее). Другой вариант: самому написать модель такого генератора. Пример модели приведен в листинге 3.
- На рис. 26 показан вид окна Virtuoso Schematic Editor, когда создание схемы будет закончено. Закроем это окно.
- Создадим вид конфигурации для ячейки test_bench («File/New/Cell View…», «Tool – Hierarchy-Editor»). В открывшимся окне New Configuration в поле «View» выберем **schematic**. Далее вызываем хранилище кнопкой «Use Template», выбираем «Name – spectreVerilog» и закрываем оба окна, дважды нажав **OK**.

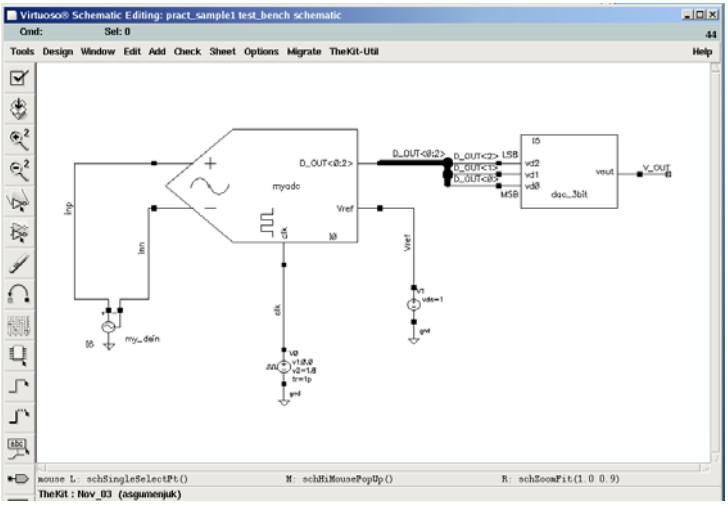


Рис. 26. Окно редактора принципиальных схем Virtuoso Schematic Editor с введенной схемой тестирования АЦП

- Откроется окно редактора конфигурации (рис. 27) Hierarchy Editor [16]. Сохраним и закроем окно.

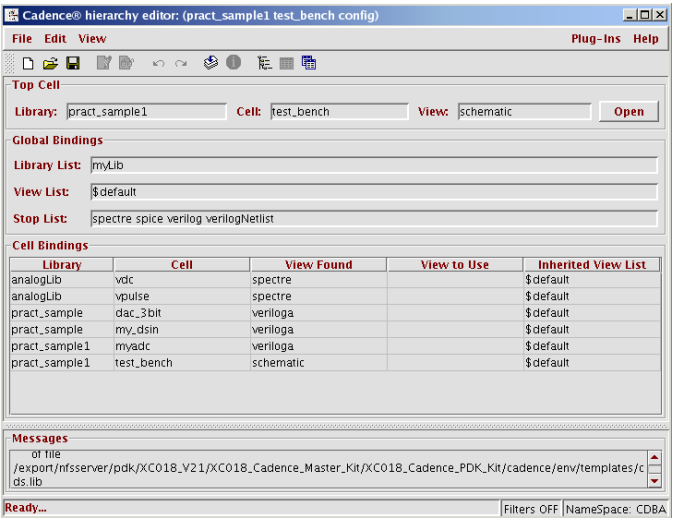


Рис. 27. Окно редактора конфигурации Hierarchy Editor

- В окне Library Manager откроем конфигурацию для моделирования, вызвав вид config ячейки test_bench. Выберем в диалоговом окне **yes – yes**, чтобы перед нами отрылись оба окна: конфигурации (см. рис. 27) и схемы (см. рис. 26).
- Теперь мы готовы к моделированию. Перейдем в окно Analog Design Environment (ADE) [17], выбрав одноименную команду в меню «Tools» окна Schematic Editor.
- Кнопкой «Choose Analyses…» выберем **tran**. Укажем время расчета «Stop Time» 41.96u. В окне Transient Options укажем параметры *strobeperiod* = 10n, *strobedelay* = 0, *skipstart* = 0.
- Попросим автоматически выводить напряжение V_OUT, выбрав меню «Outputs/To Be Plotted/Select On Schematic» ADE, а затем щелкнув по этой цепи в окне Schematic Editor.
- Создадим список цепей (netlist) и начнем моделирование, воспользовавшись меню «Simulation/Netlist and Run» или соответствующей кнопкой в правой части окна ADE.
- После завершения расчета будет открыто окно графического процессора WaveScan (рис. 28), где можно просматривать результаты расчета в графическом виде [18]. Если увеличить масштаб, то будет видна синусоида на выходе системы АЦП – ЦАП. График не будет гладким, так как наша система вносит шум квантования. Мы не видим резких переходов в виде ступенек, потому что указали симулятору (параметр *strobeperiod=10n*), что сохранять точки нужно через каждые 10 нс, а значит, на графике эти точки будут соединены прямыми линиями.
- Теперь построим спектр на выходе АЦП (сигнал, вторично преобразованный идеальным ЦАП). Это можно сделать, воспользовавшись калькулятором («Tools/Calculator» ADE) [19]. Второй вариант воспользоваться калькулятором в автономном режиме (MDL) [18]. В таком режиме калькулятор обладает немного другим набором функций, в частности функция нахождения соотношения сигнал/шум есть только в MDL-калькуляторе.
- В консоли Linux выполним команду wavescan –expr mdl.
- В окне Result Browser откроем результаты моделирования, нажав кнопку **Open**. Путь к результатам можно посмотреть и настроить в ADE («Setup/Simulator/Directory/Host…»). По этому пути находим файл test_bench/spectre/config/psf.

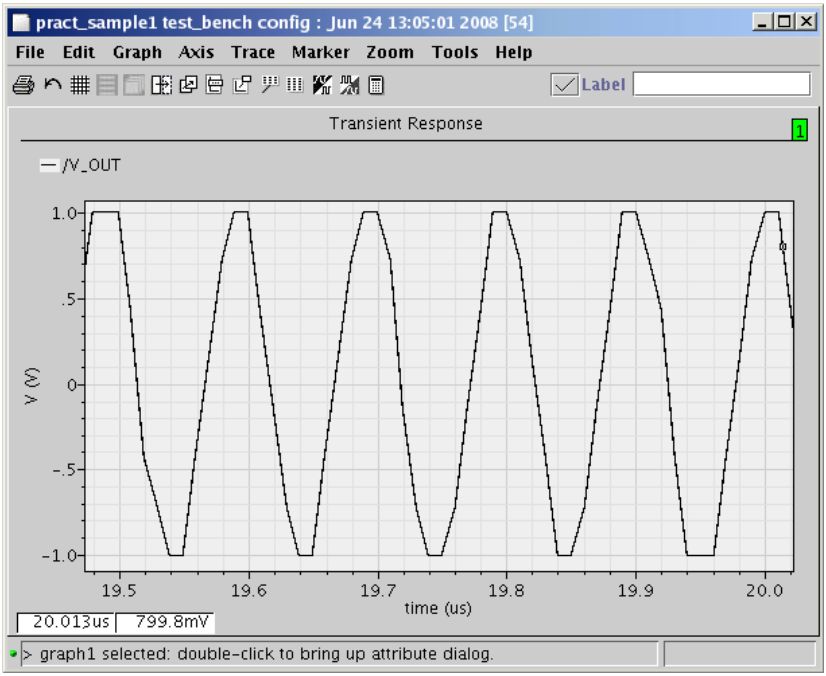


Рис. 28. Окно графического процессора WaveScan. Для просмотра выведен сигнал на выходе ЦАП

- В подкаталоге tran-tran выделим цепь V_OUT и нажмем кнопку «Signal to Calculator». Применим функцию **fft** (БПФ) со следующими параметрами: **from** = 130n, **to** = 41.09u, **numPoints** = 4096, **window** = `hanning, **smoothing** = 1.0. Будем рассматривать спектрограмму в логарифмическом масштабе – применим функцию **db**. Нажмем кнопку «plot». Полученная спектрограмма приведена на рис. 29.
- Теперь можем измерить такой динамический параметр АЦП как диапазон, свободный от гармоник SFDR. Это расстояние между основным сигналом и самым высоким на спектре бином (см. рис. 29). Пользуясь клавишей **a**, найдем это расстояние. SFDR=27.5 дБ.

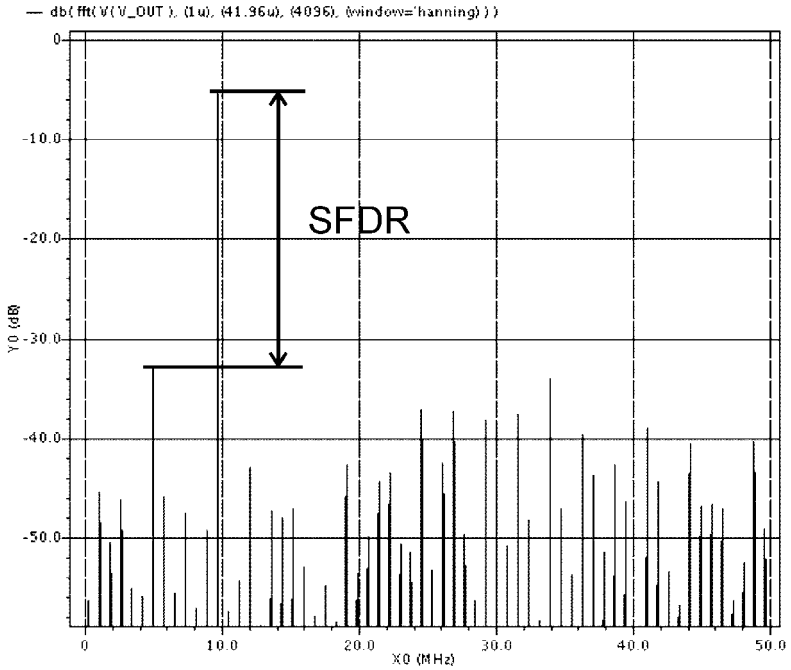


Рис. 29. Спектр сигнала на выходе АЦП при подаче на вход синусоидального напряжения максимальной амплитуды с частотой 10 МГц при частоте выборки 100 МГц

- Второй параметр SNDR, соотношение сигнал-шум плюс искажения найдем, применив к нашей функции **fft** вместо **db** функцию **snr**:

snr(fft(V(V_OUT),130n,41.09u,4096,`hanning),9.5M,10M,0,50M).

- Зная SNDR, найдем эффективную разрядность по формуле

ENOB = (SNDR – 1.76) / 6.02.

Листинг 3

```
// Verilog-A модель дифференциального
// синусоидального источника
module my_dsin (out_p, out_n);
```

```
output out_p, out_n;
electrical out_p, out_n;
parameter real freq = 1/102.4n;        // Частота сигнала
parameter real k=1;                    // Множитель частоты
parameter real offset = 0.5;           // Напряжение смещения

//Амплитуда дифференциального  напряжения
parameter real Amplitude = 0.944;

analog begin
// Функция $abstime возвращает абсолютное значение
// времени в точке расчета.
// Константа `M_PI определена файле в constants.h
V(out_p)<+ offset + 0.5*Amplitude*sin(2*`M_PI*k*freq*($abstime) );
V(out_n)<+ offset – 0.5*Amplitude*sin(2*`M_PI*k*freq*($abstime) );

end
endmodule
```

3.4.4. Отладка Verilog-A моделей

Часто возникает необходимость в отладке моделей устройств, описанных на языке Verilog-A. В версии 5.1.41 Cadence появилась удобная программа HDLdebug. Запускается она из меню «**Simulation/Netlist and Debug AHDL**» окна ADE. При этом должен быть активен нужный вам тип анализа. В открывшемся окне (см. рис. 30) можно выбрать нужный модуль (меню «**Show/Module…**»), поставить точки останова и посмотреть значения переменных.

Выводы

На этом этапе мы научились создавать схему для тестирования модели и файл конфигурации, осуществили симуляцию разработанной модели АЦП и сохранили результаты для проведения БПФ. Также научились просматривать временные диаграммы и пользоваться калькулятором в автономном режиме, узнали, как можно отлаживать Verilog-A модели, ознакомились с методикой спектрального анализа в Cadence.

Рис. 30. Окно отладчика моделей HDLdebug

Выполнив исследование идеальной модели нашего АЦП, узнали тот теоретический предел, к которому необходимо стремиться в течение всего цикла проектирования. Предельные динамические параметры, полученные на этом этапе, приведены в табл. 7.

Таблица 7

SNDR	19,9 дБ
ENOB	3,0 бита
SFDR	27,5 дБ

Самое главное, что определился общий верификационный план. Было создано тестовое окружение (система АЦП – ЦАП), в котором будет моделироваться устройство на любом уровне детализации. Был определен основной метод тестирования и измерения параметров (получение спектрограмм), который также будет использоваться в течение всего процесса проектирования.

3.5. Детализация модели АЦП

3.5.1. Схемотехническая реализация устройства

Разработанная на предыдущих этапах модель представляет собой наивысший уровень абстракции для нашего АЦП. Она очень удобна, когда данный блок используется в качестве составной части системы более высокого уровня. Но, по сути, это просто математическое описание алгоритма работы будущего устройства. Теперь нужно двигаться дальше и придумать, как реализовать эту модель в кристалле.

В соответствии с принятой концепцией проектирования сверху вниз начнем детализацию модели. На рис. 31 показан предлагаемый вариант схемотехнической реализации нашего алгоритма. Это классическая архитектура параллельного АЦП [9]. Схема АЦП разрядностью 3 бита содержит семь одинаковых компараторов напряжения I1 – I7.

Преобразуемый дифференциальный сигнал dVin поступает на один из входов всех компараторов. На второй вход этих компараторов подаются эталонные напряжения, формируемые резистивным делителем R0 – R7 из опорного напряжения Vref. Код с выходов компараторов, который часто называют температурным, подается на шифратор, который преобразует его в нужный кодовый формат, в нашем случае – в позиционный двоичный код. Температурным код называется потому, что характер изменения выходных бинарных сигналов компараторов напоминает движение столбика

жидкости по шкале термометра. АЦП всегда содержит также выходной регистр-защелку для фиксации данных, полученных в результате преобразования. В нашем примере считается, что регистр входит в состав шифратора, отсюда его название – температурный шифратор-защелка. Фиксация выходного кода осуществляется по заднему фронту тактовых импульсов на входе clk.

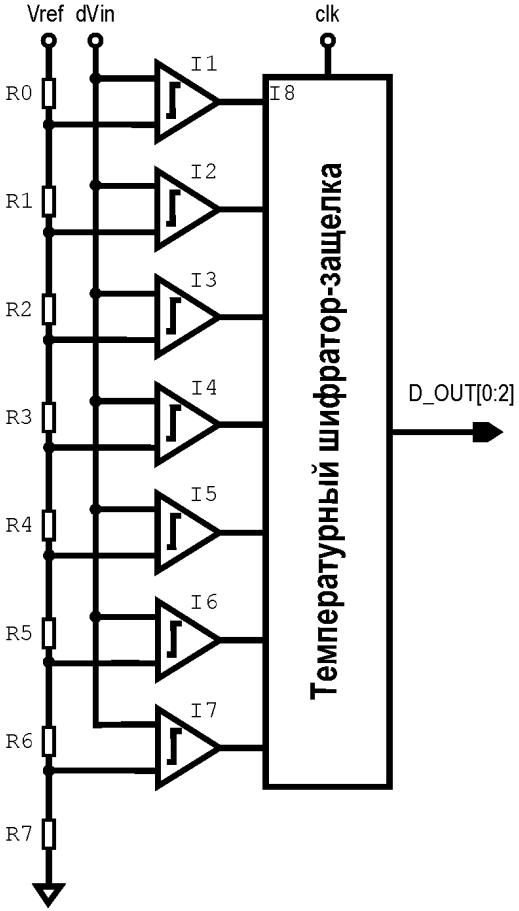


Рис. 31. Предлагаемый вариант схемотехнической реализации АЦП

В новой детализированной модели нам понадобятся компараторы, температурный шифратор и резисторы. Причем шифратор-защелка – это чисто цифровой блок, который будет разрабатываться в отдельном цифровом цикле. Возникает ситуация, когда в модели присутствуют как аналоговые, так и цифровые блоки, и для ее моделирования потребуется смешанный симулятор spectreVerilog. Выбирая архитектуру компаратора, учтем следующие требования.

- Разрешение и напряжение смещения компаратора некритичны в нашем приложении, так как разрядность АЦП небольшая и напряжение, соответствующее шагу квантования, составляет всего 250 мВ.
- Быстродействие, в свою очередь, очень важно. При частоте дискретизации 100 МГц необходимо, чтобы время установления сигнала на выходе компаратора не превышало 5 нс.

Этим требованиям удовлетворяют динамические компараторы, основанные на использовании регенеративных схем типа триггера-защелки [20]. Их работу иллюстрирует рис. 34.

Компаратор имеет два дифференциальных входа. Преобразуемый разностный сигнал по двухпроводной шине подается на выводы inp и inn, а разностное опорное напряжение также по двухпроводной шине подается на выводы refp и refn (рис. 32, *а*). Компаратор осуществляет сравнение двух дифференциальных сигналов, один из которых dVin = inp – inn, а другой dVref = refp – refn.

Сигнал стробирования, управляющий фиксацией данных, подается на вход защелкивания latch. Компаратор имеет два дифференциальных выхода: неинвертирующий, который подключен к выводу outp, и инвертирующий, подключенный к outn. Упрощенно описывая принцип работы компаратора, можно отметить следующее. Передний фронт сигнала latch определяет момент сравнения сигналов dVin и dVref (рис. 32, *б*). Значение на выходе зависит только от значений входных напряжений в момент сравнения и сохраняется в течение высокого уровня clk. Когда уровень clk – низкий, на выходе компаратора напряжение может быть любым. Это связано с применяемыми в таких узлах схемами, которые обсудим далее.

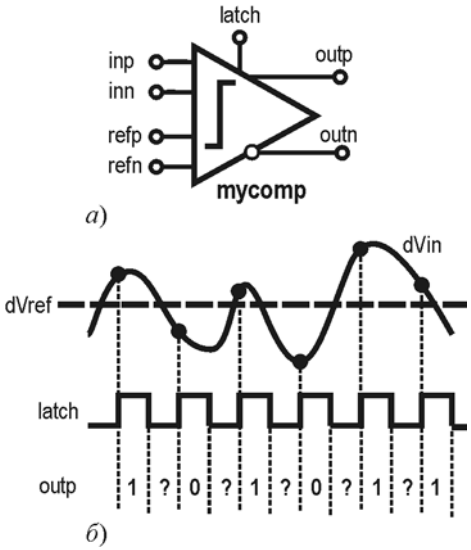


Рис. 32. Условное обозначение (*а*) и иллюстрация принципа работы (*б*) динамического компаратора разностных сигналов

Модель рассматриваемого компаратора представлена в листинге 4. В модели учитывается напряжение смещения, входная емкость и разрешение, т.е. минимальная разница между dVin и dVref, при которой компаратор принимает правильное решение. Выходное напряжение при низком уровне сигнала clk задается случайным образом с помощью системной функции $random.

Листинг 4

```
// Verilog-A модель динамического компаратора
module mycomp(outn, outp, inn, inp, latch, refn, refp);
//
output outn;
electrical outn;
output outp;
electrical outp;

input inn;
```

```
electrical inn;
input inp;
electrical inp;

input latch;
electrical latch;

input refn;
electrical refn;
input refp;
electrical refp;

// Описание параметров
parameter real log_high = 1.8;// Уровень «1»
parameter real log_low = 0;    // Уровень «0»

// Времена фронтов и задержка на выходе
parameter real out_rise_fall = 1n from [0:inf);
parameter real out_delay = 0.1n from [0:inf);

// Входное напряжение смещения
parameter real inp_offset = 0;

// Разрешение
parameter real resolution = 10u from (0:inf);
// Входная емкость
parameter real inp_cap=50f from [0:inf);
// Порог срабатывания триггера-защелки
parameter real latch_treshold=0.9;

// Декларация переменных
 real delta; //Разность дифференциального входного напряжения
//и дифференциального опорного напряжения

integer s;
real voutp, voutn; //Выходные напряжения

        analog begin
```

```
        @ (initial_step) begin // Проверка при инициализации

          if (log_high <= log_low) begin
            $display("Range specification error: out_high must be larger
then out_low");
            $finish;
          end
        end

        @( cross(V(latch) - latch_treshold, +1) or
          initial_step) begin
        // Момент защелкивания компаратора
          delta=V(inp)+inp_offset-V(inn)-( V(refp)-V(refn) );

          if (abs(delta)<resolution) begin
        // Если входная разность меньше разрешения,
        // на выходе случайные значения +1 В или -1 В
            s=100;
            delta = ( ($random(s)<50) ? 1 : -1 );
          end
        end

        @( cross(V(latch) - latch_treshold, -1 )) begin
        // Момент разблокирования компаратора
// Компаратор принимает случайное решение
          s=100;
          delta = ( ($random(s)>50) ? 1 : -1 );
        end

        if (delta >= 0) begin
        // на выходе «1»
          voutp=log_high;
          voutn=log_low;
        end
else begin
        // на выходе «0»
          voutp=log_low;
          voutn=log_high;
```

```
        end

      V(outp)<+transition(voutp,out_delay,out_rise_fall);
      V(outn)<+transition(voutn,out_delay,out_rise_fall);

        // моделирование входной емкости по каждому порту
        I(refp) <+ inp_cap*ddt(V(refp));
        I(refn) <+ inp_cap*ddt(V(refn));

        I(inp) <+ inp_cap*ddt(V(inp));
        I(inn) <+ inp_cap*ddt(V(inn));

    end

endmodule
```

Второй составной блок нашей модели, температурный шифратор-защелка, достаточно просто описывается на языке Verilog HDL (листинг 5). Имея такой мощный инструмент, как смешанное моделирование, можем не писать «аналоговую» модель на Verilog-A, а моделировать блок сразу на Verilog HDL.

Листинг 5

```
// Модель температурного шифратора-защелки
// на языке Verilog HDL
module temp_dc(c_outp, clk, d_out);
input   [0:6]    c_outp;
input            clk;
output  [0:2]    d_out;
reg [0:2] d_out;

always @(negedge clk)
   case (c_outp)

        7'b0_000_000 : d_out = 3'b000;
        7'b0_000_001 : d_out = 3'b001;
        7'b0_000_011 : d_out = 3'b010;
```

```
        7'b0_000_111 : d_out = 3'b011;
        7'b0_001_111 : d_out = 3'b100;
        7'b0_011_111 : d_out = 3'b101;
        7'b0_111_111 : d_out = 3'b110;
        7'b1_111_111 : d_out = 3'b111;

    endcase

endmodule
```

3.5.2. Практическая работа в Cadence. Создание модели смешанного типа

Создадим в библиотеке pract_sample ячейку mycomp с видами veriloga (модель из листинга 4) и symbol. Теперь займемся блоком на Verilog.

- В библиотеке pract_sample создадим новую ячейку типа verilog под названием temp_dc. Для этого в поле «Tool» окна Create New File установим Verilog-Editor, а в поле «View Name» вместо functional введем verilog.
- В окне редактора введем текст модели на Verilog HDL из листинга 5. Проверка синтаксиса и создание вида symbol для Verilog-блоков аналогична блокам на Verilog-A. Закрывая окно редактора, создадим вид symbol.
- Теперь добавим в ячейку myadc еще один вид schematic. Откроем myadc – symbol. Выберем пункт меню «Design/Create Cellview/From Cellview…» окна Symbol Editing. В результате последующего диалога будет создан вид schematic с размещенными портами.
- Соберем на основе этой заготовки схему, как показано на рис. 33. Модели резистора и контакта к общей шине («земля») можно взять в библиотеке analogLib. Номинал резисторов – 1 кОм. Сохраним схему кнопкой «Check and Save».

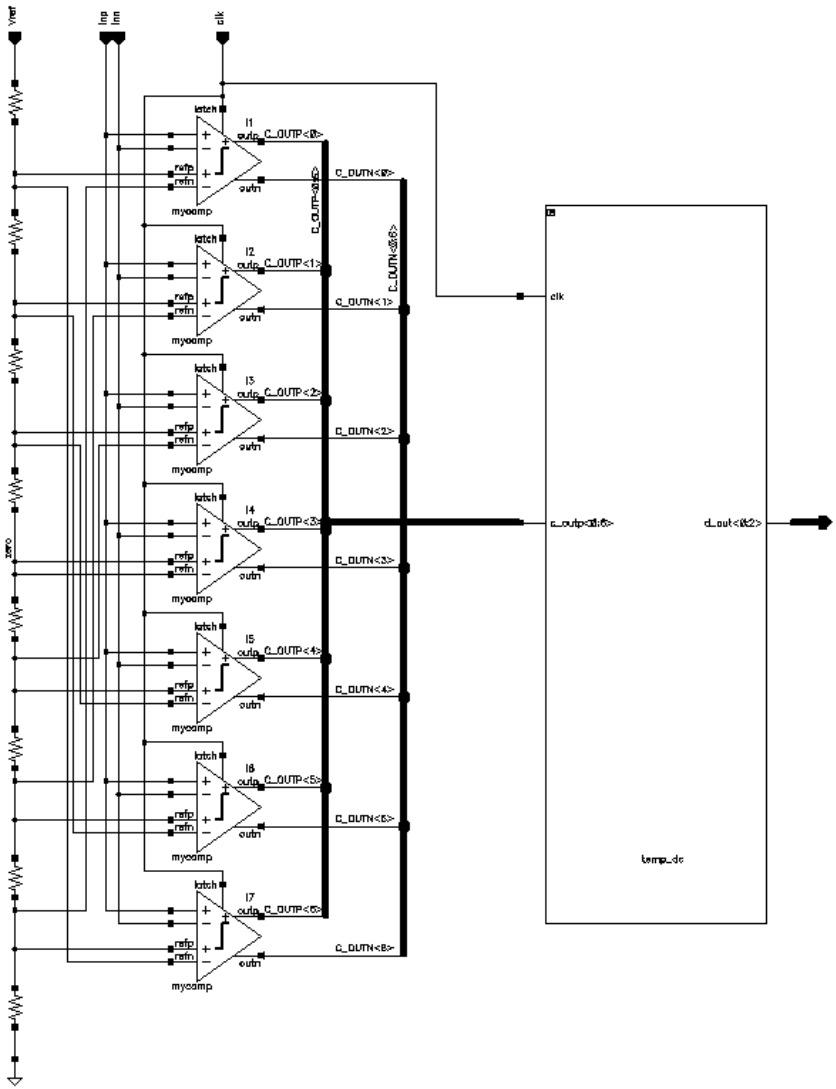


Рис. 33. Модель АЦП смешанного типа: блоки компараторов описаны на языке Verilog-A, блок шифратора – на Verilog HDL

Выводы

На этом этапе была детализирована модель. Она постепенно начинает терять свою чисто математическую сущность, приобретая черты будущего законченного изделия. Фактически разработана архитектура нашего АЦП, которую разбили на блоки. Теперь необходимо провести верификацию и сформулировать требования к блокам – компараторам и цифровой части.

3.6. Верификация архитектуры

Предлагается следующий план верификации.

- Действуя в соответствии с общим планом, получить спектр и сравнить его с результатом, полученным для математической модели.
- Посмотреть влияние напряжения смещения компараторов на динамические характеристики АЦП. Сформулировать требования к компараторам.
- Исследовать влияние таких погрешностей, как фазовый шум генератора тактовых импульсов и технологическое рассогласование резисторов.

3.6.1. Особенности выполнения смешанного моделирования средствами Cadence

Среда проектирования Cadence Virtuoso предоставляет возможности разработчику моделировать проекты, содержащие как аналоговые, так и цифровые компоненты. Смешанный компоновщик (netlister) создает сразу аналоговый и цифровой списки соединений. Для смешанного моделирования можно воспользоваться одним из симуляторов: SpectreSVerilog, spectreVerilog, UltraSimVerilog, hspiceVerilog, cdsSpiceVerilog [21]. Мы будем применять только spectreVerilog. Использование этого симулятора накладывает следующие ограничения.

- Необходимо предварительно разбить схему на аналоговую и цифровую часть.
- Проект должен содержать, по крайней мере, один аналоговый и один цифровой компонент.
- В схеме должен быть хотя бы один интерфейс.

- Цифроаналоговый интерфейс в отличие от аналого-цифрового интерфейса обеспечивает возможность установки высокоимпедансного состояния на выходе.
- Аналоговые воздействия, описываемые в файле, не могут применяться к цифровым блокам. В таком случае обязательно использовать структурные элементы, например vpulse из analogLib.
- Схема не должна содержать элементов с «подвешенными» выводами.

Интерфейсом в смешанном моделировании называют цепь, соединяющую порты аналогового и цифрового блока. На схеме (в редакторе Virtuoso Schematic) интерфейс выглядит как обычное межсоединение. Однако при компоновке списка соединений к входным и выходным портам цифровых блоков подключаются интерфейсные элементы, как показано в увеличенном фрагменте на рис. 34.

Интерфейсные элементы представляют собой специальные структуры, которые включаются между аналоговыми и цифровыми портами. Они обеспечивают:

- моделирование нагрузки и управление импедансом цифровых портов;
- преобразование уровней напряжения в логические уровни и наоборот;
- передачу информации между аналоговым и цифровым симуляторами.

Набор параметров и поведение интерфейсных элементов описывается в специальных моделях. Эти модели представляют собой эквивалентные входные и выходные схемы для цифрового блока. Существует множество различных вариантов таких моделей, учитывающих специфику различных семейств логических элементов [20].

Нас будет интересовать только стандартная модель «MOS», которая используется Cadence по умолчанию. Эта модель состоит из двух самостоятельных частей, каждая из которых применяется в зависимости от типа интерфейса. Первая, входная или *A2D*, используется для интерфейсов типа аналог-цифра. Вторая, выходная или *D2A*, – для интерфейсов типа цифра-аналог.

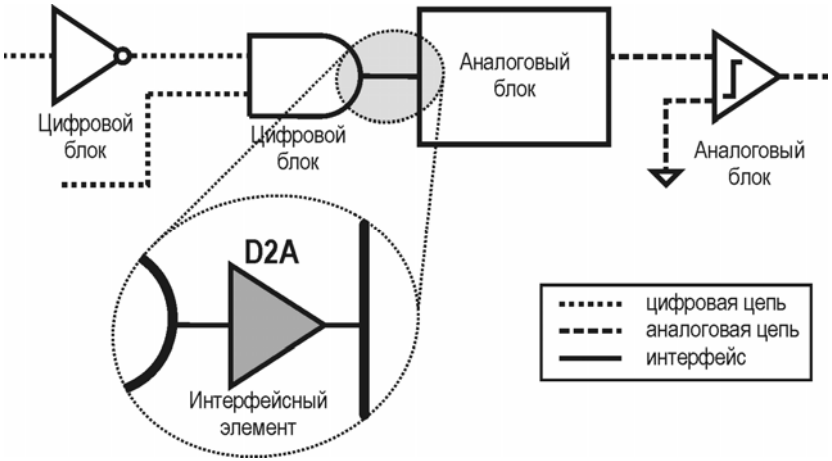


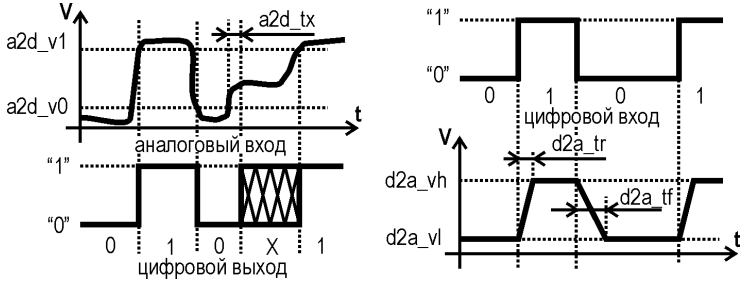
Рис. 34. Назначение интерфейсных элементов в аналого-цифровой модели

Управление интерфейсными элементами осуществляется через меню «Mixed Signal/Interface Elements» окна Schematic Editing. Настройки можно применять к отдельному порту, всему блоку сразу, ячейке или даже целой библиотеке цифровых компонентов. Параметры модели интерфейсного элемента типа «MOS» приведены в табл. 8.

Назначение параметров поясняется на рис. 35. Особого внимания заслуживает параметр a2d_tx. Фактически он регламентирует допустимую длину переходных процессов аналогового сигнала. Из рис. 35, *а* видно, что, когда время нарастания превышает величину a2d_tx, интерфейсный элемент порождает неопределенное состояние на входе цифрового блока. Это один из «подводных камней» смешанного моделирования, способный доставить массу неприятностей в виде необъяснимых ошибок при расчете цифровой части проекта. Следует быть особенно аккуратным на стадиях разработки, когда в схеме происходит стыковка блоков в транзисторном исполнении с блоками на языке Verilog, так как длительности фронтов в этом случае могут не соответствовать параметру a2d_tx, а их форма приводить к возникновению нежелательных состояний.

Таблица 8

Параметр	Описание
Входная A2D-модель	
a2d_v0	Нижний порог A2D-модели. Если напряжение в аналоговой цепи опускается ниже a2d_v0, то на вход цифровой части поступает состояние «0»
A2d_v1	Верхний порог A2D-модели. Если напряжение в аналоговой цепи превышает a2d_v1, то на вход цифровой части поступает состояние «1»
A2d_tx	Максимальное время, в течение которого напряжение в аналоговой цепи может находиться между уровнем a2d_v0 и a2d_v1 до того, как на вход цифровой части поступит неопределенное состояние «x»
Выходная D2A-модель	
d2a_vl	Напряжение на выходе цифрового порта, соответствующее состоянию "0". Если в цифровой цепи возникает состояние "0", то выходное напряжение линейно изменится до величины d2a_vl. Время перехода определяется параметром d2a_tf
d2a_vh	Напряжение на выходе цифрового порта, соответствующее состоянию "1". Если в цифровой цепи возникает состояние "1", то выходное напряжение линейно изменится до величины d2a_vh. Время перехода определяется параметром d2a_tr
d2a_tf	Время спада эквивалентного источника D2A-модели
d2a_tr	Время нарастания эквивалентного источника D2A-модели



а) Параметры A2D-модели *б) Параметры D2A-модели*

Рис. 35. Параметры аналого-цифрового (*а*) и цифроаналогового (*б*) интерфейсных элементов

Пример возможной ошибки приведен на рис. 36. Здесь рассмотрено два варианта преобразования сигнала интерфейсным элементом типа аналог-цифра. В первом варианте, когда верхний порог задан на уровне a2d_v1, на входах цифровой схемы несколько раз возникает неопределенное состояние, хотя физически такой переходной процесс не создает подобного состояния. Во втором варианте проблема была решена. Для этого напряжение порога было уменьшено на величину Δ и установлено на уровне a2d_v1'.

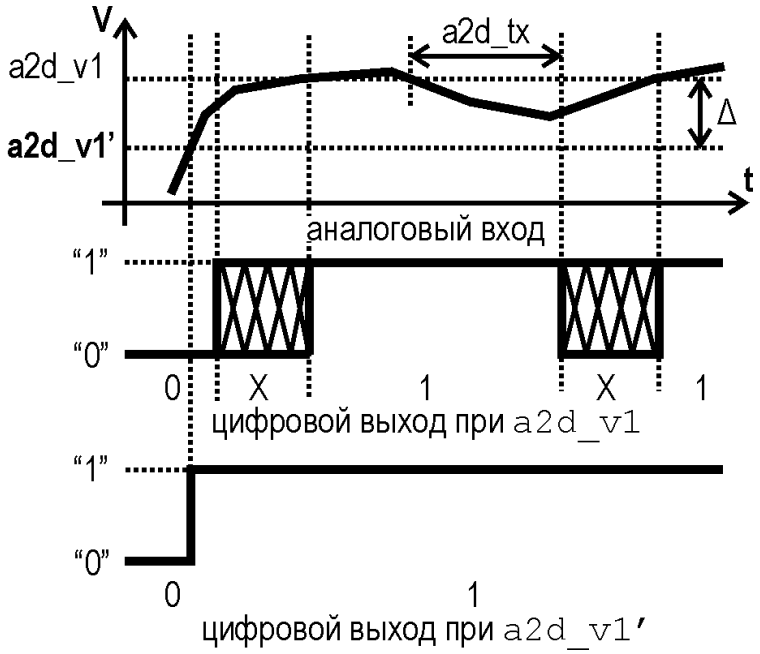


Рис. 36. Пример, показывающий возможность возникновения ошибки моделирования в случае неправильного выбора параметров аналого-цифрового интерфейса

Списки часто встречающихся ошибок, а также возможных решений проблем, возникающих при смешанном моделировании, приведены в табл. 9.

Таблица 9

Ошибка	Возможные решения
Возникновение нежелательных состояний логических элементов после преобразования интерфейсом типа аналог-цифра	Подобрать значение параметров a2d_v1, a2d_v0 близко к порогу переключения входного цифрового элемента. Увеличить значение параметра a2d_tx. Установить на вход цифрового блока буфер, описанный на Verilog-A. Тогда a2d_v1, a2d_v0 можно установить равными потенциалам питания и «земли», а параметр a2d_tx задать минимальным. Пример буфера приведен в листинге 6
Цифровой сигнал после преобразования интерфейсом не соответствует аналоговому эквиваленту, например, всегда имеет одно и то же логическое значение	Проверить параметры интерфейса a2d_v1, a2d_v0, a2d_tx. Если сигнал – внешний, т.е. интерфейс встраивается между портом схемы и входом цифрового блока, то, возможно, проблема из-за ошибки в иерархии описаний. Помочь может дополнительный аналоговый элемент, установленный между портом и цифровым блоком, например обычный резистор или буфер (листинг 6)
Проект не разбивается на аналоговую и цифровую часть (partition error)	Посмотреть сообщение об ошибке в главном окне CIW и в log-файле. Проверить настройки окружения. Перекомпилировать цифровые ячейки (открыть текст вида verilog, изменить любой комментарий и вновь закрыть)
Ошибка «Segmentation fault» с указанием модуля	Переименовать модуль. Например, если модуль назывался BUF_dig, то можно назвать его BUF_dig1
Не запускается отладчик цифровой части SimVision	Проверить скрипты, устанавливаемые с пакетом проектных библиотек. Они могут блокировать запуск отладчика. Проверить номер версии IUS. Отладчик, например, может не запускаться, если версия IUS не 5.6, а 5.7

Листинг 6

```
// Verilog-A модель цифрового буфера

module BUF_dig(out, A);

output out;
electrical out;

input A;
electrical A;

parameter real log_low = 0;
parameter real log_high = 1.8;
parameter real rise_fall_time = 1p;
parameter real td = 0;

real c, log_tres;

analog begin
        @ (initial_step) begin     // Проверка при инициализации
           log_tres=0.5*(log_low+log_high);
           c=0;

           if (log_high <= log_low) begin
             $display("Range specification error: out_high must be larger
then out_low");

             $finish;
           end
         end

        @( cross( V(A) - log_tres, +1 ) ) c = log_high;
        @( cross( V(A) - log_tres, -1 ) ) c = log_low;

        V(out)<+transition(c,td,rise_fall_time);
end
endmodule
```

3.6.2. Анализ влияния погрешности смещения компараторов

Выбранные нами компараторы динамического типа характеризуются большой величиной погрешности смещения нуля. Начнем верификацию системы с того, что определим максимально допустимое значение этого параметра, при котором параметры АЦП удовлетворяли ли бы требованиям технического задания.

Напряжение смещения или напряжение смещения нуля – это сдвиг точки переключения на передаточной характеристике компаратора относительно номинального значения (рис. 37, *а*). Причиной его возникновения является несимметричность или разбаланс схемы. При правильном размещении элементов схемы на кристалле систематическая составляющая погрешности может быть сведена практически к нулю. Случайная составляющая сохраняется и определяется технологическим разбросом компонентов. При моделировании будем исходить из того, что топология компаратора выполнена правильно, а значит, учитывать будем только случайный разброс напряжений смещения. Можно считать, что эта случайная величина распределена по нормальному закону (рис. 37, *б*) с нулевым средним при отсутствии систематической составляющей погрешности. Параметром такого распределения является величина среднеквадратического отклонения σ. Известно, что при нормальном распределении случайной величины все ее значения укладываются в диапазон ±3σ с вероятностью более 99,9 %.

С учетом этого внесем изменения в модель так, чтобы напряжение смещения компараторов подчинялось нормальному закону распределения. Тогда моделирование позволит выяснить, при каких максимальных значениях σ параметры АЦП удовлетворяют ТЗ. Если величина σ окажется достаточно большой, то отсюда можно сделать вывод, что схема допускает значительный разброс параметров компонентов. Следовательно, архитектура компаратора выбрана верно, и можно проводить ее дальнейшую детализацию.

Дополним модель нашего компаратора так, чтобы напряжение смещения подчинялось нормальному закону. Измененный текст модели компаратора приведен в листинге 7. Точнее в нем приведены только новые фрагменты кода, дополняющие текст модели из листинга 4. Строки из точек показывают расположение фрагментов исходного текста модели.

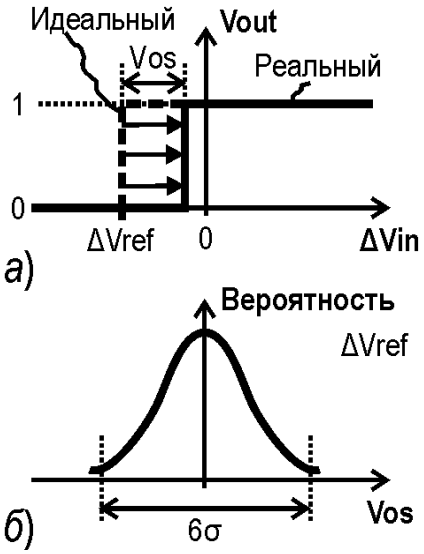


Рис. 37. Передаточная функция компаратора (*а*) и распределение случайной составляющей напряжения смещения нуля (*б*)

Язык Verilog-A предусматривает формирование нормально распределенных случайных величин. Для запуска генератора случайных (точнее, псевдослучайных) чисел необходима стартовая точка, задаваемая специальным параметром. Однако это не означает, что последовательность каждый раз будет одинакова при постоянном стартовом числе. На самом деле, если стоят два генератора с одинаковым стартовым числом, то каждый раз при запуске симулятора оба генератора выдадут одинаковые числа, но эти числа не будут равны числам из предыдущей симуляции. Если поставить 8 компараторов с 8 разными стартовыми числами, то каждый раз при запуске симулятора будут генерироваться 8 различных и отличных от предыдущих чисел.

Дописав модели, и задав для каждого компаратора свой параметр inp_offset_seed и одинаковый для всех параметр inp_offset_std, сможем промоделировать АЦП со случайным распределением напряжений смещения, причем с каждым следующим запуском си-

мулятора последовательность будет иной. Этот факт можно использовать для моделирования случайного разброса и анализа по методу Монте-Карло.

Чтобы решить для себя вопрос, какая же максимальная величина σ является допустимой, потребуется сделать как можно более широкую выборку. В этом учебном проекте для каждого значения σ нам предстоит сделать по четыре измерения, хотя в условиях реального проектирования этого конечно мало.

Листинг 7

```
// Модель компаратора со случайным смещением
module mycomp_model(outn, outp, inn, inp, latch, refn, refp);

output outn;
electrical outn;
……...
input refp;
electrical refp;

// Описание параметров
parameter real log_high = 1.8;
……...
parameter real out_delay = 0.1n from [0:inf);

parameter real inp_offset_std = 0;// СКО напряжения смещения

// Параметр генератора случайного смещения
parameter integer inp_offset_seed = 221;

parameter real resolution = 10u from (0:inf);
……

// Декларация переменных
real delta;
integer s;
real voutp, voutn;
real inp_offset;
```

```
integer seed;
// Описание функционирования модели
        analog begin
          @ (initial_step) begin
          // Проверка при инициализации

            if (log_high <= log_low) begin
              $display("Range specification error: out_high must be larger
then out_low");

    $finish;
            end

        // Задаем начальную точку генератора случайных чисел
            seed=$random % inp_offset_seed;

// Получаем распределение смещения по нормальному закону
// при СКО=inp_offset_std
   inp_offset=$rdist_normal(seed,0,inp_offset_std);

// Выводим информацию на экран
            $display("offset =",inp_offset);

          end
……….
        end
endmodule
```

Требования к компараторам. Окончательно определившись с архитектурой, можем сформулировать требования к компараторам. При определении требований к их быстродействию надо учесть следующее. Максимальное время между передним и задним фронтами импульсов синхронизации на входе clk составляет 5 нс. Из этого интервала необходимо вычесть длительность фронтов и время удержания, в течение которого данные на входе триггера не могут меняться. В сумме это составит приблизительно 2 нс с запасом

на разброс временных параметров. Тогда требования к компараторам можно сформулировать в следующем виде:

- архитектура – дифференциальная, динамического типа [20];
- время срабатывания – 3 нс;
- диапазон входных синфазных сигналов – от 0 до 1 В;
- напряжение питания – (1,8 ± 0,2) В.

3.6.3. Влияние технологического разброса компонентов

К сожалению, все элементы интегральных микросхем подвержены влиянию разброса технологических параметров. Следствием этого является отклонение абсолютных значений характеристик компонентов от их номинальных величин и относительное взаимное рассогласование характеристик групп компонентов. Поэтому невозможно добиться полного совпадения параметров двух одинаковых элементов на пластине: транзисторов, резисторов или конденсаторов. Рассогласование может иметь случайный или систематический характер. Причиной первого является случайный разброс технологических параметров. Среди причин систематического рассогласования можно отметить механические напряжения, возникающие при установке кристалла в корпус и заливке его пластмассой, неравномерное распределение температуры по пластине (градиентный эффект) и др. Краткая сводка причин рассогласования:

- случайный разброс технологических параметров;
- градиенты температуры, уровней легирования и других характеристик рабочих слоев чипа;
- эффект близости – влияние близко расположенных структур на параметры компонентов;
- анизотропные эффекты – зависимость физических характеристик кристалла от ориентации пластины.
- Ниже приведен ряд рекомендаций, направленных на улучшение согласования элементов.
- Следует по возможности увеличивать размеры элементов, прежде всего ширину и длину каналов МОП транзисторов. Понятно, что чем больше размер элемента, тем в меньшей степени сказывается разброс при его изготовлении. Например, степень рассогласования по какому-либо параметру пары МОП транзисторов ΔP (разность значений порогового напряжения или крутизны) харак-

теризуется величиной среднеквадратического отклонения этого параметра, которая определяется выражением:

$$\sigma^2(\Delta P) \sim A_p^2 /(W \cdot L) + D_x \cdot S_p^2,$$

где A_p – эмпирическая константа, определяемая разбросом площади канала; S_p – константа, определяемая влиянием эффекта «близости»; D_x – расстояние между транзисторами. Влияние дистанции начинает сказываться только на больших расстояниях (свыше 500 мкм), поэтому основной вклад в рассогласование транзисторной пары вносит фактор, зависящий от площади затвора.

- В токовых зеркалах длина каналов транзисторов должна быть одинаковая.
- Транзисторы должны работать в режиме сильной инверсии.
- Крупные компоненты или группы компонентов следует разбивать на небольшие фрагменты и располагать их, следуя специальным правилам, так, чтобы сформировать массивы элементов с общим центром [9, 22 – 24]. Они должны обладать симметрией относительно градиентов температуры на кристалле. Взаимное расположение элементов должно быть таким, чтобы чередовались фрагменты разных компонентов из одной группы.

Топологические методы улучшения согласования компонентов сведены в табл. 10.

Таблица 10

	Причины рассогласования компонентов			
Методы улучшения согласования	Случайный разброс	Градиенты	Эффект близости	Анизотропия
Минимизация пустот		X		
Увеличение размеров транзисторов	X			
Близкое параллельное размещение		X		X
Окружение фиктивными элементами			X	
Разбиение элементов на фрагменты, помещаемые в массивы с общим центром	X	X		
Одинаковая ориентация стоков и истоков МОПТ				X

В нашем АЦП наибольший вклад в статическую погрешность преобразования вносит рассогласование номиналов резисторов делителя напряжения. Для выявления чувствительности схемы к разбросу сопротивлений резисторов проведем моделирование случайной вариации их номиналов. Идея следующая: по аналогии с компараторами заменим все стандартные резисторы своими моделями, в которых отклонение от номиналов подчиняется нормальному закону. Цикл модельных испытаний позволит определить среднеквадратическое отклонение от номинала, при котором заметно ухудшаются параметры АЦП. Текст модели резистора приведен в листинге 8.

Листинг 8

```
// Модель резистора, учитывающая случайный разброс номиналов
module res(vp, vn);
inout vp, vn;
electrical vp, vn;
parameter real R_mean = 1K;     // среднее значение номинала
parameter real R_std = 0.000;     // СКО
parameter integer R_seed = 773; // начальная точка генератора
real Res, dRes;
integer seed;

  analog begin

        @(initial_step) begin

          seed=$random % R_seed;
          dRes=$rdist_normal(seed,0,R_std)*R_mean;
          Res=R_mean+dRes;
          $display("Res =",Res);
        end

    I(vp, vn) <+ V(vp, vn)/Res;

  end
endmodule
```

3.6.4. Исследование влияния апертурной неопределенности

Существенный вклад в общую погрешность АЦП вносят шумы. К ним часто относят и так называемый джиттер (jitter), или фазовый шум, который является причиной особого вида динамической погрешности АЦП – апертурной неопределенности, т.е. отсутствия точной временной привязки момента взятия выборки. Джиттер – это случайное дрожание фронта сигнала, которое может быть вызвано различными факторами: собственными фазовыми шумами тактового генератора, помехами в линии передачи и др. Джиттер специфицируется как временной параметр, характеризующий величину среднеквадратического отклонения от средней величины фазовой задержки. В своей модели мы учтем влияние случайного дрожания фронта синхросигнала clk. Влияние апертурной неопределенности на параметры АЦП известно. В частности, максимальное значение отношения сигнал-шум (SNR) при частоте входного сигнала f_S и величине джиттера Δt ограничивается на уровне

$$\mathrm{SNR} = -20\cdot\log(2\pi f_S \Delta t).$$

Вклад этой погрешности при частоте входного сигнала 10 МГц сравним с шумом квантования трехбитного АЦП, если величина джиттера составляет около 1,8 нс. Проверим это с помощью следующей модели. Между источником тактового сигнала и входом АЦП установим блок джиттер-генератора. Принцип функционирования блока поясняется на рис. 38.

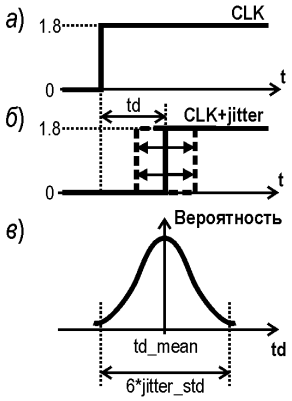


Рис. 38. Принцип функционирования джиттер-генератора

Очередной фронт сигнала clk задерживается на время td, задаваемое каждый раз случайно в соответствии с нормальным законом распределения. Параметры распределения: среднее значение времени задержки – td_mean, среднеквадратическое отклонение – jitter_std. Другими словами, это просто цифровой буфер со случайной задержкой. Текст модели джиттер-генератора приведен в листинге 9.

Листинг 9

```
// Модель джиттер-генератора
module jitter_del(inp, out);

input inp;
electrical inp;

output out;
electrical out;

parameter real V1 = 0;
parameter real V2 = 1.8;
parameter real trise = 200p;
parameter real tfall = 200p;
parameter real td_mean = 500p;
parameter real jitter_std = 0;
parameter integer jitter_seed = 173;
real vout;
real vtrans;
real td;
integer seed;

analog begin

        @ (initial_step) begin
                vtrans = (V1 + V2)*0.5;
                vout = ( V(inp)<vtrans ? V1 : V2 );
                td = td_mean;
                seed=$random % jitter_seed;
        end
```

```
        @( cross( V(inp) - vtrans, +1) )
          begin
                vout = V2;
                td=abs($rdist_normal(seed,td_mean,jitter_std));
          end

        @( cross( V(inp) - vtrans, -1) )
          begin
                vout = V1;
                td=abs($rdist_normal(seed,td_mean,jitter_std));
          end

        V(out) <+ transition( vout,
                  td, trise, tfall );
end

endmodule
```

3.6.5. Практическая работа в Cadence. Смешанное моделирование

Прежде чем перейти к практическому освоению способов смешанного моделирования, необходимо освоить основной прием правильной методологии проектирования – научиться управлять конфигурацией проекта. Конфигурирование позволяет легко и удобно переходить с одного уровня детализации на другой. В основе конфигурирования лежит сама структура базы данных Cadence – набор ячеек и видов. Собирая схему модели или тестового окружения, мы пользуемся символами ячеек, а их внутреннее содержание выбирает конфигуратор.

Посмотрим на окно конфигуратора, показанное на рис. 27. Таблица «Cell Bindings» содержит все ячейки, входящие в состав блока верхнего уровня, сведения о котором можно видеть в рамке «Top Cell» в верхней части окна. Напротив каждой ячейки (графа «Cell») указан тот вид, который используется в текущей конфигурации. Поскольку до сих пор у нас было только по одному подходящему виду (schematic или veriloga), то могли вообще не пользоваться конфигуратором.

Теперь в ячейке myadc к виду veriloga добавился новый вид schematic, содержащий схему детализированной модели. Чтобы использовать его вместо первоначально установленного вида, нужно изменить конфигурацию.

Любой имеющийся вид можно установить, выбрав его в контекстном меню «Set Cell View», которое вызывается нажатием правой кнопки мыши по соответствующей строке в таблице ячеек. Для обновления конфигурации выполним следующие действия.

- Откроем окно Cadence hierarchy editor. С помощью контекстного меню укажем конфигуратору вместо вида veriloga ячейки myadc использовать вид schematic.
- Обновим конфигурацию, нажав кнопку «Update». В таблице должны появиться новые строки, содержащие ячейки детализированной модели: mycomp, temp_dc, res.
- Бывает полезно выделить в схеме виды, используемые в каждом блоке. Для этого в меню Schematic Editor выберем пункт «Hierarchy-Editor/Show Views Found…». Если такой пункт меню отсутствует, воспользуйтесь меню «Tools/Hierarchy Editor», тогда он должен появиться.

Следующим шагом настроим смешанный режим моделирования [21]. Еще раз оговоримся, что необходимость смешанного моделирования возникает вследствие наличия как аналоговых, так и цифровых блоков. Появление в проекте блока шифратора, описанного на языке Verilog HDL, не позволяет дальше моделировать систему обычным симулятором spectre. Продолжим работу.

- Сменим симулятор на spectreVerilog. С помощью меню «Setup/Simulator/Directory/Host» окна ADE откроем окно соответствующих настроек. В списке «Simulator» выберем пункт «spectreVerilog». Закроем окно кнопкой OK.
- В меню «Tools» окна Schematic Editing выберем пункт «Mixed Signal Opts.». В строке главного меню появится соответствующий пункт.
- Вызовем окно Partitioning Options, выбрав одноименный пункт в меню «Mixed-Signal». Убедимся, что в поле «Analog Stop View Set» присутствует veriloga, а в поле «Digital Stop View Set» – verilog.

Теперь весь проект может быть разбит на цифровую и аналоговую части. Убедиться в этом можно, выделив все блоки и межсоединения разными цветами, в зависимости от принадлежности к одному из классов. Делается это с помощью меню «Mixed-Signal/Display Partition/All Active».

- Настроим параметры интерфейсов блока temp_dc. Для этого перейдем на пункт меню «Mixed-Signal/Interface Elements/Cell…» окна Schematic Editor, а затем щелкнем левой кнопкой мыши по пиктограмме блока temp_dc.
- В открывшемся окне IE Model Property Editor в списке «Model IO» выберем «input». Значение параметров следующие: a2d_v1=1.7, a2d_v0=0.1, a2d_tx=1n. Указав их в соответствующих полях, нажмем кнопку Apply.
- Теперь в списке «Model IO» выберем «output». Повторим процедуру для параметров: d2a_tf=1n, d2a_tr=1n, d2a_vh=1.8, d2a_vl=0. Закроем окно кнопкой OK.
- Чтобы избежать ошибок при использовании интерфейсов в иерархических цепях, а таковой у нас является цепь clk, добавим в нее резистор номиналом 1 Ом, как показано на рис. 39. Созданный при этом аналого-цифрой интерфейс будет включен между резистором и входом clk блока temp_dc, а не между портом clk и цифровым узлом.
- Выполним моделирование новой схемы уже в смешанном режиме, повторив последовательность действий из раздела «Тестирование модели АЦП».
- Посмотрев спектр, убедимся, что детализированная модель работоспособна, а ее параметры близки к параметрам идеальной математической модели. У нас получился практически такой же спектр, а значения параметров АЦП соответствуют приведенным в табл. 5. После настройки моделирования в смешанном режиме займемся исследованием влияния случайного разброса напряжений смещения.
- Подготовим еще один блок компаратора по новой модели из листинга 7. Ячейка будет размещаться в библиотеке pract_sample и называться mycomp_model. В ее состав войдут виды veriloga и symbol.

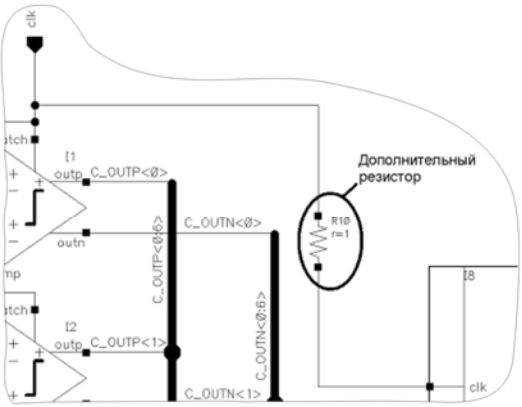


Рис. 39. Способ исключения возможных ошибок при наличии интерфейсов в иерархических цепях

- Создадим новый вид в ячейке myadc. В окне Library Manager скопируем вид schematic ячейки myadc в вид schematic_model той же ячейки.
- Заменим все блоки mycomp в схеме myadc.schematic_model на новые компараторы mycomp_model.
- В свойствах каждого из восьми компараторов новой схемы установим индивидуальный параметр inp_offset_seed, например 10, 20, 30, … Параметр inp_offset_std зададим глобально, указав в свойствах глобальную переменную offset_std.
- Конфигуратору укажем использовать вид schematic_model вместо schematic для ячейки myadc. Обновим конфигурацию.
- Далее, проводя временное моделирование при разных значениях глобальной переменной offset_std (задается в окне ADE), получим набор спектров. Измерения следует повторять по 4 раза для каждого значения переменной. Результат будет разный, так как имеет место случайный разброс напряжений смещения компаратора. Полученные результаты занесем в таблицу. В качестве примера в табл. 11 приведены данные, полученные в одном из циклов моделирования. Заметим, что в соответствии с требованиями ТЗ значения SNDR и ENOB должны быть не менее 16 дБ и 2,4 бит соответственно.

Таблица 11

Разброс смещения, мВ	SNDR, дБ	ENOB, бит
150	12,5	1,8
	15,0	2,2
	18,8	2,8
	13,6	2,0
125	15,2	2,2
	15,8	2,3
	17,8	2,7
	13,7	1,9
100	18,5	2,8
	16,7	2,5
	15,3	2,3
	17,9	2,7
80	18,2	2,7
	16,0	2,4
	16,4	2,4
	15,6	2,3

Конечно, четырех измерений недостаточно для статистической выборки, но нам и не требуются точные данные. Полученный результат больше качественный – уже при разбросе в 100 мВ получим значения параметров АЦП, удовлетворяющие требованиям ТЗ.

Динамические компараторы имеют средний разброс напряжения смещения десятки милливольт, следовательно, их можно использовать в проекте. Проверим влияние технологического разброса резисторов.

- Создадим ячейку резистора res с видами veriloga (модель листинга 8) и symbol, который можно скопировать из библиотеки analogLib.
- В схеме myadc.schematic_model заменим все резисторы из analogLib на модели res, укажем для каждого свое стартовое число

генератора R_seed и глобально зададим одинаковый разброс R_std (для начала 0,001, т.е. 0,1 %).

- Проведем серию расчетов, проверяя влияние разброса на параметры модели.

Можно убедиться, что эффективная разрядность АЦП начинает снижаться при разбросе номиналов резисторов больше 20 %. Учитывая, что при условии правильного топологического проектирования абсолютный разброс сопротивлений резисторов составляет менее процента, можно считать, что требования ТЗ удовлетворяются.

Далее рассмотрим влияние не идеальности запускающих импульсов. Исследуем реакцию нашей модели на наличие джиттера в сигнале синхронизации.

- Создадим в библиотеке pract_sample ячейку джиттер-генератора jitter_del с видами veriloga (модель листинга 9) и symbol.
- В схеме тестового окружения test_bench установим блок jitter_del между источником тактового сигнала и входом clk АЦП. Зададим для него параметр jitter_std=110p.
- Получим спектрограмму для частоты входного сигнала, заданной выражением «16/102.4n». Результат приведен на рис. 40.

Если сравнить полученный спектр с идеальным спектром, не содержащим шума, обусловленного случайным дрожанием фронта тактового сигнала (рис. 29), то видно, что он стал более реалистичным, похожим на тот, что можно увидеть, например, на экране осциллографа.

Рассматривая спектрограмму можно также наблюдать эффект трансляции и наложения спектральных компонентов аналогового сигнала при дискретизации [14], вследствие которого частота 160 МГц оказалась в первой зоне Найквиста.

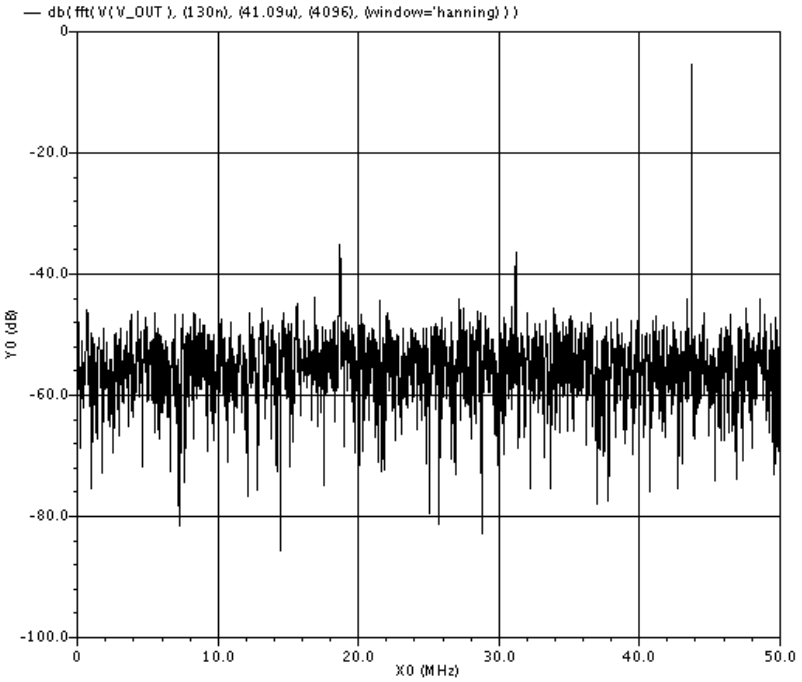


Рис. 40. Спектр выходного сигнала АЦП при наличии джиттера тактового сигнала

Отладка цифровых Verilog моделей. Для отладки цифровых блоков, описанных на языке Verilog в среде их непосредственного функционирования, удобно пользоваться отладчиком SimVision [25]. Он вызывается автоматически при запуске моделирования, если в окне опций цифрового симулятора (меню «Simulation/Options/Digital…» окна ADE) установлены элементы управления «SimVision Debugger» и «Stop After Compilation».

Как работать с отладчиком читайте в [25]. Отметим лишь, что все интересующие сигналы следует выводить в окно просмотра до начала симуляции. Моделирование следует проводить до момента времени, меньше установленного в окне ADE, иначе все результаты будут сброшены.

Выводы

В этом разделе, пользуясь проверенным тестовым окружением и в соответствии с учрежденным планом, провели верификацию модели АЦП. В результате была подтверждена правильность выбора архитектуры, как самого АЦП, так и входящих в его состав компараторов. Исследовано влияние случайных отклонений номиналов резисторов и напряжений смещения нуля компараторов, а также джиттера тактового сигнала на параметры АЦП.

3.7. Переход на транзисторный уровень

На предыдущих этапах нам удалось построить и верифицировать поведенческую модель нашего устройства. При этом не имели дело ни с технологическими библиотеками, ни с моделями конкретных элементов (транзисторов, резисторов и т.д.). Эту фазу еще называют эскизным проектом. В принципе, полностью завершить эскизный проект устройства можно, даже не определившись с технологией, по которой это устройство будет в дальнейшем разрабатываться. Поэтому выполнение всех ранее описанных заданий не должно было вызвать затруднений, даже если вообще отсутствовали какие-нибудь проектные библиотеки.

Сейчас вступаем в новый этап или фазу разработки, называемую техническим проектом. Дальнейшая детализация модели будет связана с реализацией блоков на уровне транзисторов, а для их симуляции потребуются соответствующие модели. В связи с этим нужно выбрать технологию изготовления ИМС, получить и установить комплект проектных библиотек.

3.7.1. Пакет проектных библиотек

Пакет проектных библиотек (Process Design Kit, PDK) или база данных для проектирования – дополнительный комплект файлов, обеспечивающий связь между САПР и производителем микросхем. Он включает модели и пиктограммы элементов, параметризованные ячейки, правила контроля технологических норм и экстракции параметров элементов и многое др.

Роль пакета проектных библиотек в цикле разработки поясняется диаграммой на рис. 41. На этапе разработки схемы (Front-End

design) он предоставляет модельные файлы и пиктограммы элементов: транзисторов, резисторов, конденсаторов и т.д. Пакет проектных файлов содержит также следующие данные, необходимые на этапе физической реализации (Back-End design).

- Технологический файл и набор параметризованных ячеек (PCell). Технологический файл включает в себя описание технологических слоев и соответствующие им номера масок, вид слоев в топологическом редакторе. Параметризованные ячейки – ячейки, автоматически меняющие свои виды в зависимости от значений параметров, задаваемых пользователем. Пример такой ячейки – транзистор, топология которого автоматически перерисовывается в зависимости от размеров канала. Все элементы библиотеки (транзисторы, резисторы, конденсаторы и др.) обычно поставляются в виде параметрических ячеек. Это значит, что топологию для них рисовать не нужно – она будет генерироваться сама, в зависимости от параметров компонента. Это существенно ускоряет разработку, поскольку не требует ручной прорисовки каждого транзистора или резистора.
- Правила верификации технологических норм (**DRC** – **D**esign **R**ule **C**hecks).
- Правила, регламентирующие идентификацию компонентов в топологии и экстракцию их параметров (**LPE** – **L**ayout **P**arameter **E**xtractions). Благодаря этим данным, например, программа распознает в топологии транзистор и определяет размеры его канала.
- Правила сравнения топологии со схемой (**LVS** – **L**ayout **V**s **S**chematic).
- Исходные данные для экстракции паразитных элементов топологии (**PEX** – **P**arasitic **EX**traction).

Кроме того, пакет библиотек содержит всевозможные вспомогательные программные модули (скрипты), обратные связи (callback), внушительный пакет документации и все, что должно облегчить жизнь разработчику.

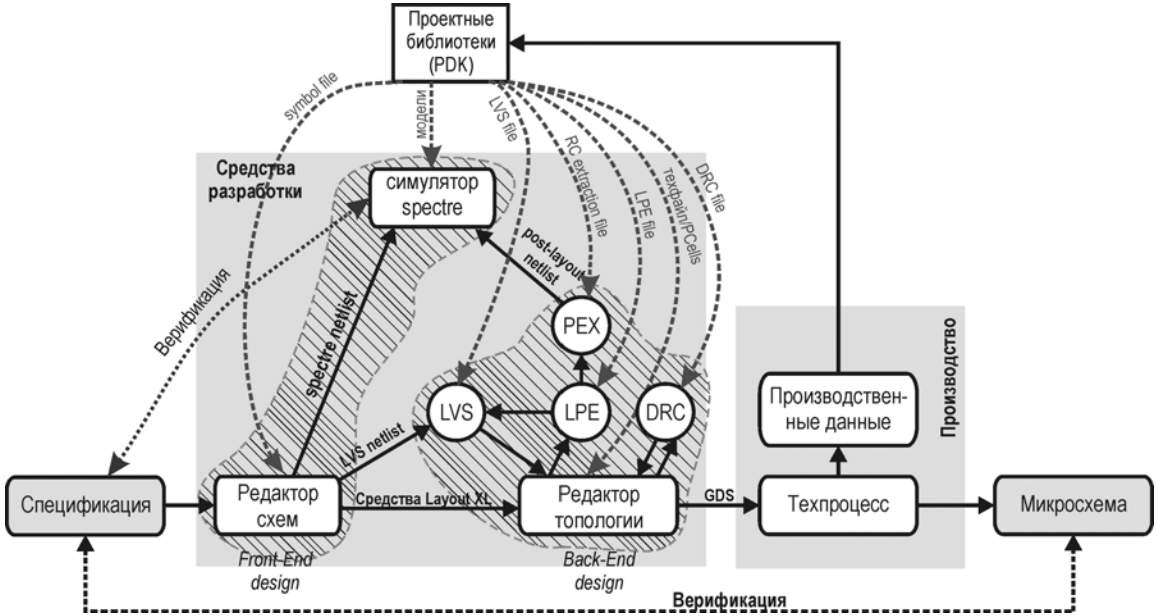


Рис. 41. Роль пакета проектных библиотек в проектировании ИМС

Дополнительно центрам проектирования ИМС предоставляются библиотеки цифровых элементов для работы с САПР цифровых схем, а также библиотеки компонентов, необходимых на окончательной стадии разработки: контактные площадки, элементы защиты от статического электричества и т. д.

Исходные данные, на которых базируется пакет проектных библиотек, – характеристики технологического процесса. На фабрике проводится многократная характеризация процесса, создаются тестовые структуры, из которых экстрагируются параметры слоев и структур. При помощи специального оборудования и программного обеспечения измеряются характеристики тестовых элементов и разрабатываются их модели. И все это делается постоянно, поскольку всегда параллельно с производством идет доводка, оптимизация техпроцесса.

Создание хорошего пакета проектных библиотек для фабрики – сложная, кропотливая и дорогостоящая работа. Вместе с тем, это – лицо фирмы, ее визитная карточка. И неслучайно, ведь он позволяет разработчику легко и быстро, не разбираясь во всех тонкостях технологии, проектировать, сосредоточившись на своих задачах, и быть уверенным, что результаты компьютерного моделирования соответствуют физическим реалиям.

В этом учебном проекте можно использовать различные пакеты проектных библиотек, например таких производителей, как XFAB, UMC. В данном случае применяется база данных для проектирования XFAB. Это библиотека элементов, реализуемых по КМОП технологии с проектными нормами 0,18 мкм.

После установки проектных библиотек можно приступать к детализации модели. Прежде всего, требуется разработать динамический компаратор в соответствии с требованиями, сформулированными в предыдущем разделе.

К сожалению, в данном пособии не сможем подробно рассмотреть методику разработки схемы компаратора. Об этом можно прочитать в соответствующих статьях или книгах. Возможный вариант схемы с напряжением питания 1,8 В показан на рис. 42. Нас интересует, как эту схему включить в цикл проектирования законченной ИМС, в данном случае АЦП.

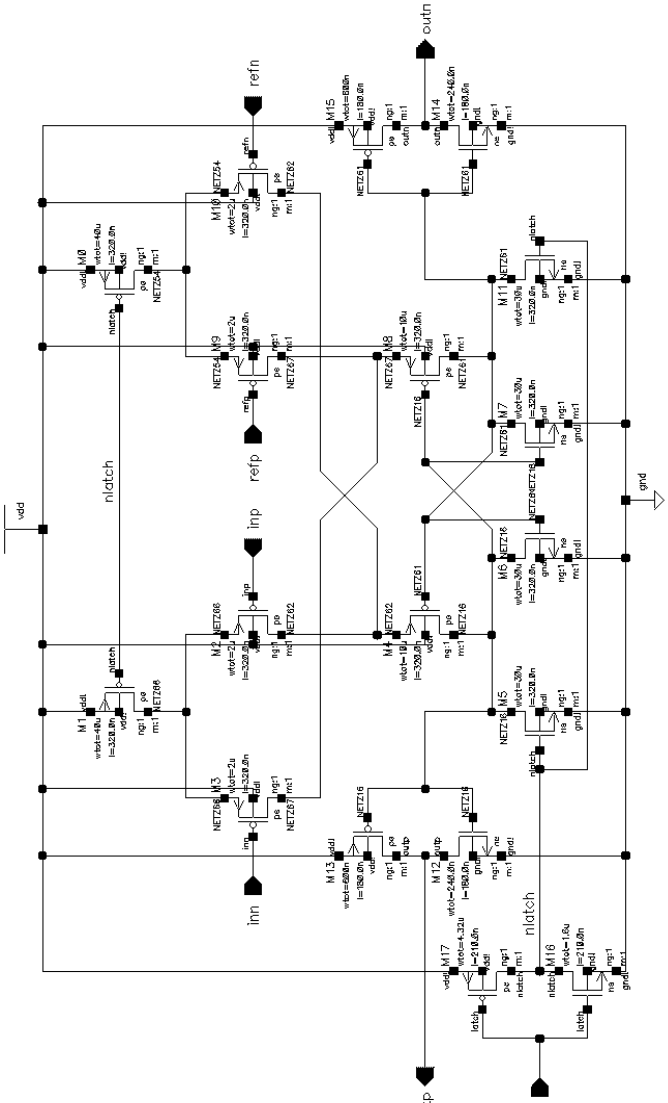


Рис. 42. Принципиальная схема компаратора

При проектировании следует всегда иметь в виду наличие возможных отклонений параметров технологического процесса. Этот разброс учитывается в моделях элементов: транзисторов, конденсаторов и т. д. Подключая модель, указываем секцию параметров. Таких секций всегда минимум три: «медленная», когда пороговые напряжения низкие и значения крутизны транзисторов минимальны; «быстрая», когда, наоборот, значения крутизны максимальные; и «типовая», когда все параметры принимают среднее значение.

В зависимости от схемы разработку проекта можно вести на типовых моделях, а потом протестировать на остальных или же сразу выбрать худший вариант. Заметим, что для аналоговых схем совсем необязательно худшими будут «медленные» транзисторы.

В любом случае, если нужно получить нормальный выход годных микросхем, следует добиваться хороших результатов верификации при всех вариациях параметров моделей, включая экстремальные варианты для используемого технологического процесса.

3.7.2. Практическая работа в Cadence. Моделирование на уровне транзисторов

Используем предложенную схему динамического компаратора на КМОП транзисторах в нашем проекте. Порядок дальнейших действий следующий.

- Создать вид schematic у ячейки mycomp. Выполнить ввод схемы, показанной на рис. 42.
- Создать тестовое окружение для компаратора: ячейка mycomp_test с видами schematic и config. Верифицировать окружение можно, используя Verilog-A модель компаратора. Отладить компаратор в этом окружении.
- Тестировать АЦП, подставив вид schematic компаратора вместо veriloga. Добиться результатов, удовлетворяющих ТЗ.
- Повторить все шаги для каждого из вариантов моделей КМОП транзисторов (медленный, типовой и быстрый).
- Повторить все шаги, проверяя схему на работоспособность в указанном в ТЗ температурном диапазоне.
- В результате получим спектр выходного сигнала АЦП при синусоидальном входном воздействии подобный тому, который показан на рис. 43.

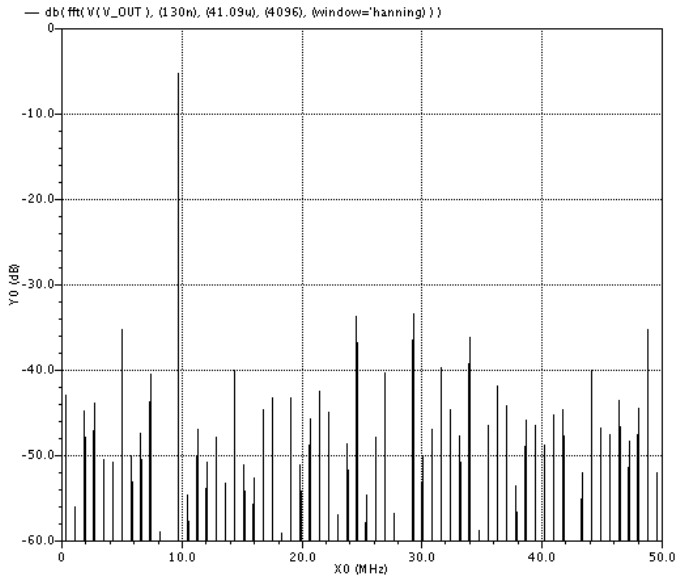


Рис. 43. Спектр выходного сигнала АЦП, в котором модели компараторов представлены на транзисторном уровне, а модель шифратора описана на языке Verilog

Еще один неохваченный детализацией элемент – шифратор, описанный на языке Verilog. Этот блок разрабатывается в отдельном цикле с помощью специальных инструментов. Он должен быть синтезирован в базисе цифровых компонентов проектной библиотеки. Уже упоминалось, что в состав пакета проектных библиотек должны входить цифровые элементы, и сейчас они понадобятся. Цифровая библиотека в Cadence – совокупность ячеек и видов, разделенных на категории: сумматоры, инверторы, буферы, триггеры и др. Причем ячейки в каждой категории отличаются своей нагрузочной способностью, о чем можно судить по специальным суффиксам в обозначении. Например, у буферов в библиотеке XFAB имеются суффиксы X1, X2, X3, …, X20. Обычно элементы разрабатываются так, чтобы нагрузочная способность равнялась четырем равнозначным элементам. Скажем, суффикс X8 означает, что буфер может быть нагружен на 4 элемента X8 или на 16 элементов X4 или на 2 элемента X16. Но все это условно, ведь специальные

САПР при синтезе сами подбирают элементы с учетом их нагрузочной способности, обеспечивая требования разработчика.

Каждая ячейка в цифровой библиотеке содержит стандартный набор видов. Обычно в него входят symbol, cmos_sch (эквивалент schematic), layout (топология для синтеза), functional (или verilog) и еще несколько специфических видов, например dataSheet. Стандартные виды позволяют использовать цифровую библиотеку и в своих разработках, если это требуется. Скажем, если нужен инвертор, необязательно разрабатывать свой, можно использовать стандартный.

Все характеристики блоков в цифровой библиотеке описаны в файле специального табличного формата. В этом файле перечислены задержки элементов при различных нагрузках и длительностях входных фронтов. Как в моделях транзисторов и конденсаторов, существует секция медленных и быстрых режимов. Этот табличный файл требуется при синтезе и предназначен для САПР цифровых схем. Вот информация, которая нужна для синтеза нашего цифрового блока:

- максимальная задержка логического элемента – 5 нс;
- емкость нагрузки (только внутренней шины) – 0,5 пФ;
- длительности фронтов входных сигналов – 1 нс.

Результатом синтеза будет так называемый нетлист или список цепей, т.е. список элементов и связей между ними, в который будут включены компоненты из цифровой библиотеки. Этот нетлист можно импортировать в Cadence, преобразовав его в вид schematic. Пример списка цепей, полученного в программе Cadence RTL Compiler после синтеза дешифратора, приведен в листинге 10. Он представляет собой описание связей компонентов цифровой библиотеки. Например, BULX3 – это КМОП буфер с нагрузочной способностью 3. Строки из точек помещены в тех местах, где должны находиться фрагменты списка цепей, исключенные из листинга с целью сокращения его объема.

Листинг 10

```
// Generated by Cadence RTL Compiler (RC) 05.20-p002
module temp_dc(c_outp, clk, d_out);
input   [0:6]    c_outp;
```

```
input          clk;
output [0:2]   d_out;
wire n_2, n_3, n_4, n_5, n_6, n_9, n_10, n_11;
……
wire n_44, n_45;

BULX3 n0000D(.A (n_5), .Q (n_31));
NA2LX1 n0001D(.A (n_35), .B (n_37), .Q (n_38));
ON21LX1 n0001D128(.A (n_24), .B (c_outp[3]), .C (n_20),
 .Q (n_37));
NA2LX1 n0001D134(.A (c_outp[1]), .B (c_outp[0]), .Q (n_5));
BULX8 p0000A(.A (n_38), .Q (d_out[1]));
……
endmodule
```

Импортируем синтезированный нетлист в формат Cadence.

- В основном окне icfb CIW выберем пункт меню «File/Import/Verilog…».
- В открывшемся окне настроим параметры импорта. В поле «Target Library Name» укажем имя библиотеки, в которую будет осуществлен импорт. Можно указать pract_sample, но желательно всегда импортировать в новую библиотеку, чтобы не засорять рабочую. В поле «Verilog Files To Import» вносим имя файла с нетлистом. В поле «Reference Libraries» – имя цифровой библиотеки. Если требуется, указываем имена шин «земли» и питания в поле «Global Nets».
- После нажатия кнопки **OK** в указанной библиотеке возникнет ячейка temp_dc с видом schematic, состоящим из элементов цифровой библиотеки. Если импорт не состоялся, то одной из причин может быть неверно указанная (или не указанная совсем) переменная окружения LD_LIBRARY_PATH.

Теперь появляется возможность проверить работу дешифратора на транзисторах в схеме АЦП. Для этого следует в конфигурации указать для него вид schematic. Цифровые элементы в его составе должны быть представлены на транзисторном уровне.

Наконец, чтобы убедиться в функциональности полной схемы, на заключительном шаге выполнения проекта следует заменить все

идеальные резисторы их моделями из проектной библиотеки (выберем поликремневый резистор *p*-типа) и затем промоделировать весь АЦП на транзисторном уровне, не забыв учесть температуру и технологический разброс. Моделирование теперь следует проводить симулятором spectre, а не spectreVerilog, так как в модели отсутствует цифровая часть.

Выводы

В этом разделе мы приступили к фазе выполнения технического проекта – реализации устройства по конкретной технологии. Нам удалось разработать схемы составных блоков АЦП – компаратора и дешифратора. Если аналоговый блок компаратора был спроектирован вручную, путем тщательного подбора размеров транзисторов, то схема дешифратора была синтезирована с помощью САПР цифровых устройств.

В результате была подготовлена и верифицирована полная схема устройства. Несмотря на то, что на следующем этапе она будет дополнена некоторыми элементами, да и разработанные схемы блоков могут быть скорректированы после создания топологии, тем не менее, первый основной этап разработки, так называемый Front-End design, можно считать завершенным. Результаты моделирования АЦП на транзисторном уровне и сравнение их с результатами, полученными ранее, приведены в табл. 12.

Таблица 12

Уровень абстракции моделей	SFDR, дБ	SNDR, дБ	ENOB, бит
Модель на поведенческом уровне	28	19,9	3,0
Модель на транзисторном уровне («типовые» МОП транзисторы)	29	19,8	2,99
Модель на транзисторном уровне («медленные» МОП транзисторы)	28	19,9	3,00
Модель на транзисторном уровне («быстрые» МОП транзисторы)	28	19,7	2,98

3.8. Проектирование топологии основных блоков

3.8.1. Введение

Проектирование топологии или физическая реализация – наиболее трудоемкая часть проекта. Работа тополога совсем не похожа на работу разработчика: тополог не создает схему, он, пользуясь своими знаниями в области физики, технологии и конструирования микроэлектронных компонентов, реализует ее на кристалле. Роль тополога заключается в подготовке эскизов для масок фотолитографии, которые будут применяться на фабрике в процессе изготовления ИМС. Эта работа весьма специфична, требует специальных знаний, особых навыков и опыта, поэтому в крупных центрах проектирования микросхем ею занимаются отдельные подразделения.

Если давать определение топологическому проектированию, то можно сказать, что это – процесс создания аккуратного физического представления схемы, отвечающего требованиям как технологии производства, так и ТЗ на изделие. Под требованиями технологии понимаются различные ограничения и правила расположения элементов, их минимальные и максимальные размеры, требования, связанные с обеспечением надежности и т. д. Эти ограничения являются следствием чрезвычайной сложности процесса изготовления микросхемы.

Топологический рисунок – просто совокупность геометрических фигур в различных слоях. Эти слои могут иметь отношение к реальным слоям физической структуры микросхемы. Например, слой ME1 соответствует первому металлу. Вместе с тем, слой может быть абстрактным, не имеющим физической реализации. Например, слой TEXT представляет собой специальный слой для технических обозначений на топологии. Законченный топологический рисунок микросхемы отправляется на фабрику, где на его основе происходит генерация масок для фотолитографии. Маски формируются применением к слоям топологического рисунка различных функций: какие-то маски полностью эквивалентны рисунку в определенном слое, какие-то формируются инверсией или пересечением, т.е. логическим перемножением слоев. Но тополога интересуют сами топологические слои, потому что работать ему приходится с ними, а не с масками. Описания слоев, правила и ограничения, ко-

торых необходимо придерживаться при их использовании, всегда приводятся в проектных библиотеках.

В качестве примера рассмотрим топологию и поперечное сечение p-канального МОП транзистора (рис. 44). Для создания топологии использованы следующие шесть слоев:

- NWELL – слой, на основе которого будет создан *n*-карман;
- MET1 – слой первого металла;
- POLY1 – слой поликремния;
- PIMP – слой *p*+ имплантации;
- DIFF – слой диффузии или активная область;
- CONT – контакт к активной области.

Реально маска для вскрытия окон получается пересечением трех слоев: CONT, DIFF, PIMP.

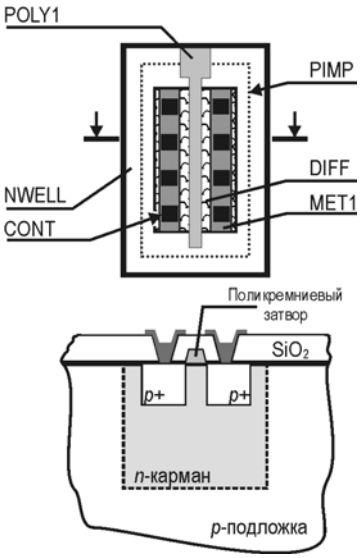


Рис. 44. Упрощенное изображение топологии и поперечного сечения *p*-канального МОП транзистора

Обратите внимание на различие между топологией и технологической реализацией: на топологическом рисунке слой PIMP содержит один большой прямоугольник, а реально на кристалле образу-

ются две отдельные *p*+ области, разделенные затвором. Объясняется это тем, что с целью получения самосовмещенной структуры сначала делается затвор, а затем имплантация. Затвор действует как экран, под который имплантация не проникает, а образует две отдельные области.

Слой DIFF, называемый диффузионным слоем или активной областью, представляет собой окно вскрытия в толстом оксиде (Field Oxide, FOX), в котором затем наращивается тонкий диэлектрический слой. Этот слой является вспомогательным, поскольку обретает смысл только в местах пересечения его другими слоями. Так, имплантация производится в область, образованную пересечением слоев PINP (или NIMP) и DIFF. Поэтому, если, например, выполнить охранное кольцо (*p*+ имплантация) просто слоем PINP, забыв DIFF, то при изготовлении чипа ионы не будут имплантироваться в подложку, а будут попадать в толстый изолирующий оксид, не проникая сквозь него. В свою очередь, пересечение поликремния и диффузии означает, что в этом месте должен быть затвор. Это пересечение определяет ширину и длину затвора транзистора.

Удельное сопротивление слоя поликремния составляет около 200 Ом/квадрат, что существенно больше, чем у слоя металлизации – приблизительно 0,1 Ом/квадрат. Учитывая, что паразитная емкость у дорожки поликремния больше, чем у металлической дорожки, не следует чрезмерно увлекаться разводкой в слое поликремния.

Выбранный пример наглядно показывает, что рисунки топологических слоев не являются изображениями технологических масок. Обратите также внимание, что слой подложки отсутствует. Это значит, что все пространство на топологическом чертеже и есть подложка.

При топологическом проектировании не создаются также маски для межслойных диэлектрических разделительных пленок. Маски для таких служебных слоев будут сформированы на фабрике автоматически путем применения различных логических операций к уже имеющимся слоям. Учитывая, что маска требуется для каждой технологической операции, число их обычно больше числа доступных в топологическом редакторе слоев.

3.8.2. Этапы проектирования топологии

Типовой маршрут разработки топологии следующий [26]:

- создание топологического плана – предварительная оценка площади и размещение блоков;
- поблочная разработка – размещение и трассировка компонентов с учетом требований плана;
- верификация топологии – визуальная и с применением САПР;
- экстракция паразитных параметров и моделирование.

Рассмотрим подробнее каждый из этих этапов.

Подготовка топологического плана. Топологический план представляет собой эскиз будущей топологии микросхемы. Размещение блоков на кристалле является предварительным, пока полностью не завершено топологическое проектирование каждого из них.

При подготовке плана действуют следующим образом:

- размещают шины питания и глобальные сигнальные шины;
- располагают входные и выходные интерфейсы (контактные площадки);
- создают эскиз предварительного размещения блоков с учетом их особенностей (наличия симметрии в расположении элементов, необходимости защиты от шума по подложке и от эффекта защелкивания);
- оценивают занимаемую площадь.

Следует учитывать сопротивление шин питания, подводимых к каждому блоку. Аналоговые схемы более чувствительны к отклонениям напряжения питания, чем цифровые узлы, да и потребляемый ток у них обычно больше, чем у цифровых схем не очень высокой степени интеграции. Поэтому способы подведения шин к аналоговым и цифровым блокам – различные. На рис. 45, *а* показан способ подключения шины питания (или шины «земли») к цифровому ядру: контактная площадка подключается к кольцу, от которого уже напряжение питания подается на ядро. На рис. 45, *б* показан способ подключения шины питания к аналоговым блокам. Из-за большого потребления тока использовать кольцо нельзя, так как вдали от контактной площадки этот ток, протекая по кольцу, вызовет значительное падение напряжения. Это приведет к тому, что в

разных точках подключения питания к цифровой части уровни питающих напряжений будут отличаться, а в некоторых цепях они могут оказаться недопустимо низкими. Поэтому от контактной площадки «землю» и питание разводят «звездой», так что к каждому блоку они подключатся индивидуальной шиной. Дополнительным преимуществом такой трассировки является значительное ослабление взаимных помех между блоками, возникающих из-за протекания возвратных токов по общей шине. В связи с отмеченной особенностью подключения питания к аналоговым и цифровым блокам следует уже на начальном этапе определить расположение контактных площадок, используемых для подачи питания и соединения с шинами «земли». Заметим, что таких контактных площадок, как для подключения питания (аналогового и цифрового), так и для подключения «земли» (аналоговой и цифровой) может быть несколько. Их количество обычно ограничено числом доступных выводов корпуса ИМС.

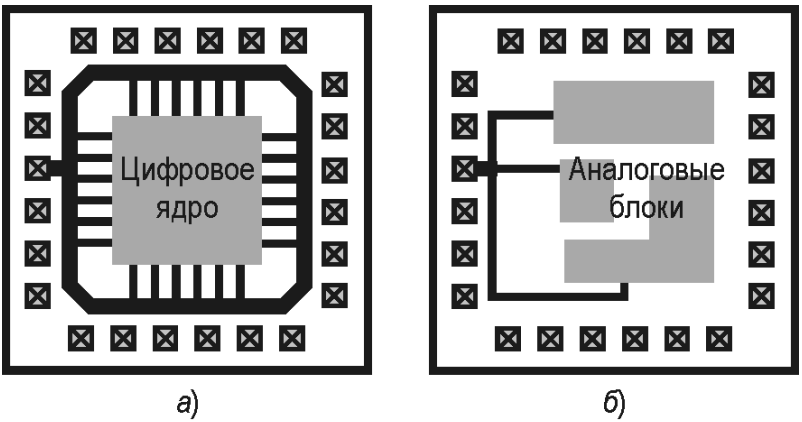


Рис. 45. Способы подключения шин питания к цифровым (*а*) и аналоговым (*б*) блокам ИМС

Поблочная разработка проекта. По аналогии с разработкой схемы топология проектируется небольшими фрагментами, которые затем объединяются в систему. Это естественно облегчает разработку, так как не приходится думать, как трассировать тысячи

транзисторов одновременно. Иерархичность проекта упрощает и верификацию, кроме того, скорость проверки DRC и LVS иерархического проекта на порядок больше чем одноуровневого. Занимаемое дисковое пространство многоуровневой топологии также намного меньше, чем неиерархической. В свою очередь, составные блоки могут быть использованы в других частях проекта или в дальнейшей работе над другими проектами. Преимущества иерархических проектов следует использовать и вести разработку топологии поблочно снизу вверх.

Верификация топологии. Это одна из важнейших стадий проектирования. Разработка в области микроэлектроники отличается от разработки в других областях одним существенным требованием: нужно *сразу* создать правильно работающее изделие. Если, например, программист может в любое время исправить неудачный код, то «забраться» в чип и заменить неисправные элементы, к сожалению, не получится. Если ошибка будет обнаружена после начала производства, даже опытного, не говоря уже о промышленном, то ее исправление потребует много времени и средств. Именно поэтому следует исправлять ошибки, пока они еще «виртуальные», уделяя максимум внимания верификации проекта с помощью САПР. Верификация подразумевает целый ряд автоматических проверок, например соответствие проекта технологическим нормам или соответствие схемы и топологии. Кроме того, важно проводить и визуальную проверку топологии, недоступную интеллекту машины, что естественно требует определенного опыта.

Экстракция и моделирование. Дорожки межсоединений, переходные отверстия, рядом расположенные структуры – все это вносит дополнительные паразитные элементы в топологию блока, изменяя его характеристики. Вместе с развитием технологии и уменьшением проектных норм происходят изменения в толщинах основных слоев. Так, межслойные диэлектрики становятся тоньше, а высота (толщина) проводящих дорожек – больше. Это приводит к изменению соотношения ширины и высоты проводящих дорожек с развитием современных технологий. Если раньше ширина проводника превышала его высоту, то теперь – наоборот.

Скажем, в технологии 0,18 мкм при минимальной ширине проводников первого металла 0,23 мкм их высота составляет около 0,5 мкм. Толщина межслойного диэлектрика при этом порядка

1,5 мкм. На рис. 46 показано, как идущие рядом две дорожки первого металла создают взаимную емкость большую, чем емкость между одной из дорожек и верхним металлом. Поэтому, проектируя топологию, следует обращать внимание не только на перекрытие металлических слоев, но и на близко расположенные проводники.

При использовании технологий с проектными нормами меньше 0,5 мкм не допустимо игнорировать влияние паразитных емкостей и сопротивлений на параметры схем. Поэтому важнейшей процедурой становится экстракция этих элементов из топологии и моделирование схемы вместе с ними.

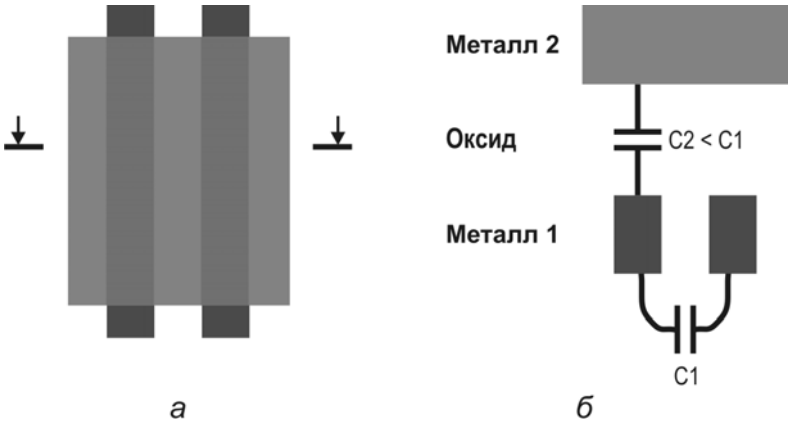


Рис. 46. Вид сверху (а) и сечение (б) двух параллельных дорожек металлизации, их взаимная паразитная емкость и емкость дорожки относительно верхнего слоя металлизации

3.8.3. Физическая реализация схемы АЦП

На предыдущих этапах постепенно разрабатывались схемы всех блоков, начиная с простейших, и каждый из них проходил верификацию совместно с другими блоками, входящими в состав АЦП. Этот же принцип разработки переносится и на физическую реализацию, только уровень схемы заменяет уровень топологии.

Уровнем топологии некоторого блока будем называть список цепей (нетлист), содержащий помимо компонентов схемы и связей

между ними также паразитные элементы межсоединений. Например, уровень топологии нашего компаратора – это нетлист, включающий список транзисторов, описание соединений их выводов, а также описание дополнительных компонентов – резисторов и конденсаторов, которые моделируют паразитные элементы межсоединений. Эти компоненты автоматически включаются в схему на этапе физической верификации и экстракции паразитных параметров.

Создание топологии в Cadence включает следующие этапы.

- *Импорт элементов* из редактора схем в редактор топологии (Virtuoso Layout XL). На основе схемы можно автоматически создать предварительную заготовку ее топологии. Для этого требуется, чтобы каждый элемент схемы имел свою топологию (по умолчанию вид layout). Уже говорилось о том, что все компоненты проектной библиотеки обычно включают этот вид, причем он параметризован, поэтому топология генерируется автоматически для каждого элемента с учетом его параметров. Следовательно, топология каждого МОП транзистора будет сформирована индивидуально.
- *Размещение элементов (placement)*. Сейчас имеются средства автоматизации размещения аналоговых компонентов, например NeoCell, но чаще всего это приходится делать вручную, в редакторе Virtuoso.
- *Трассировка (routing)* т.е. создание межсоединений. Если не применять специальных инструментов, то и этот процесс выполняется в ручном режиме. Единственное на что можно рассчитывать – это подсказки топологического редактора Layout XL. Он помогает разработчику, показывая, какие цепи подлежат соединению.
- *Проверка технологических норм (DRC)*. Это обязательная и часто единственная проверка, которой подлежат все проекты на входном контроле любой фабрики. Проверяется соответствие рисунка топологии установленным для данной технологии требованиям. Эти требования, описанные в соответствующих правилах DRC, обусловлены физическими ограничениями и особенностями процессов изготовления ИМС и поэтому являются *обязательными* для соблюдения. Отступ от этих правил, хотя бы по одному из многочисленных пунктов, снимает ответственность изготовителя за конечный результат.

- *Сравнение топологии со схемой (LVS)*. Эта проверка не менее важна, но в большей степени для разработчика, чем для производителя. Вероятность ошибки даже у опытного тополога при разводке больших проектов очень велика. Цена этой ошибки – потерянные месяцы и огромные финансовые затраты на изготовление неработающего образца. Инструмент LVS – очень мощное средство, способное многократно уменьшить вероятность ошибки. Принцип верификации LVS заключается в сравнении двух списков цепей. Первый формируется на основе исходной схемы, а второй является результатом экстракции параметров топологии (LPE), которая производится после запуска процедуры LVS. Верификатор идентифицирует в топологии примитивные элементы: транзисторы, резисторы и конденсаторы, а также их параметры: ширину, длину и т. д. Просматривается каждый объект в топологии и создается база данных компонентов и межсоединений, на основе которой создается второй список цепей. Сравнивая их между собой можно выявить ошибки трассировки и размещения: закоротки, разрывы, отсутствие в топологии необходимых элементов или связей.
- *Проверка электрических норм (ERC – electrical rules check)*. Эта операция верификации не указана на рис. 31, потому что она часто выполняется совместно с LVS. Набор правил, проверяемых ERC, описывается в соответствующих файлах проектной библиотеки. Примеры проверок, проводимых совместно с LVS: неподключенные подложки, плавающие цепи, закоротки, антенные эффекты. Существует набор важных проверок, которые часто не выполняются автоматически и которые разработчику приходится делать вручную, рассчитывая лишь на свою внимательность и опыт. Такие ошибки могут даже стать причиной отказа микросхемы, и это после того, как процедуры верификации DRC и LVS прошли успешно. Речь идет об ошибках, связанных с разводкой шин питания, соблюдением норм по электромиграции, а также с защитой выводов микросхемы от электростатического разряда. Вот пример ERC ошибки, которая часто не обнаруживается верификатором LVS: на блок, который потребляет большой ток, питание подается по тонкой шине. Последствия такой ошибки понятны – перегорание шины и выход микросхемы из строя. Существуют специальные средства контроля норм при прокладке шин питания, отслеживаю-

щие возможность просадок напряжения, а также контроля норм, относящихся к электромиграции, например Cadence VoltageStorm.

- *Экстракция паразитных параметров (RCX)*. Это заключительный этап, на котором создается нетлист схемы с паразитными компонентами. Если применяются технологии с проектными нормами меньше 0,5 мкм, то моделирование аналоговой схемы с дополнительными «паразитными» элементами следует проводить обязательно.

Описание методов физической верификации с помощью широко применяемого пакета Cadence Assura можно найти в [27].

Прежде чем приступить к физической реализации нашего устройства, подготовим топологический план (floorplan) – эскиз топологии будущего кристалла. На рис. 47 представлен вариант предварительного размещения основных узлов, контактных площадок и шин питания.

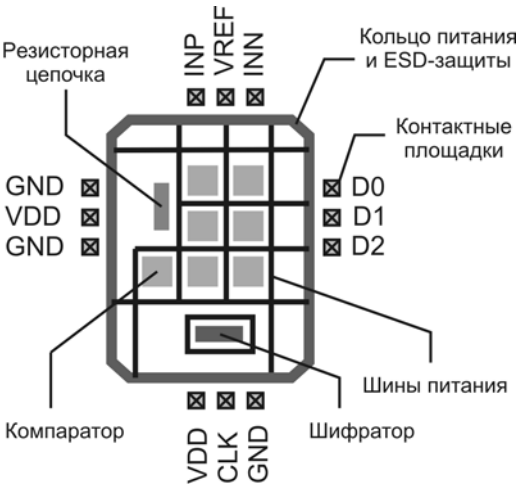


Рис. 47. Топологический план проектируемого кристалла

Для простоты на рисунке не показаны детали подключения шин питания и трассировки прочих сигнальных шин, хотя их можно нанести на расширенный, более детальный план топологии. Шины питания сделаны в виде решетки, подключаемой к кольцу, как де-

лают в цифровых схемах. В данном случае не стали прибегать к трассировке «звездой», потому что потребление наших компараторов незначительное. О кольце защиты от электростатического разряда поговорим в следующем разделе.

3.8.4. Практическая работа в Cadence. Разработка топологии компаратора

- Откроем схему mycomp.schematic. Выберем пункт «Tools/Design Synthesis/Layout XL» окна Schematic Editing. Создадим новый вид layout. Откроется окно Virtuoso XL Layout Editing и окно редактора слоев LSW.
- Перенесем все транзисторы компаратора в топологию, воспользовавшись пунктом меню «Create/Pick From Schematic…» окна XL Layout Editing. Для переноса элементов схемы в топологию их необходимо предварительно выделить на схеме.
- Подсветим на схеме линии связей с помощью окна Show Incomplete Nets, вызываемого из меню «Connectivity/Show Incomplete Nets…».
- Размещаем элементы и проводим межсоединения. Подробную информацию по работе с редактором топологии Layout XL можно найти в [28]. Проектированию топологии микросхем посвящено много книг, в частности [22 – 24]. Наш вариант топологии компаратора показан на рис. 48.
- Верифицируем топологические нормы с помощью пакета средств физической верификации Assura [27]. Верификацию можно выполнить по окончании работы над топологией, но лучше топологическое проектирование и верификацию проводить параллельно, и после нескольких изменений топологии желательно делать проверку. Так, легче искать и исправлять ошибки. Запуск верификатора DRC производится из меню «Assura/Run DRC…».
- Если ошибок в размещении элементов на топологии не обнаружено, можно проводить процедуру сравнения со схемой (меню «Assura/Run LVS»). Указав в соответствующих полях имена видов топологии и схемы, запускаем верификатор. По окончании проверки Assura выведет диалоговое окно, с помощью которого можно в интерактивном режиме исправить возникшие ошибки. Не надо пугаться, если число этих ошибок в первый раз будет очень боль-

шим. Дело в том, что верификатор осуществляет формальное сравнение схемы и топологии по нескольким признакам, поэтому одна небольшая ошибка может вызвать сразу ряд нарушений различных правил. Поэтому, число ошибок разрастается как снежный ком. Зато, обнаружив, исправив с помощью интерактивной среды Assura одну ошибку и повторив процедуру LVS, можно увидеть, что список сообщений об ошибках значительно поредел.

- Когда верификатор LVS выдал сообщение о том, что схема соответствует топологии, приступаем к экстракции. Выберем пункт «Assura/Run RCX…» в меню Virtuoso Layout XL.

- Получив вид av_extracted, есть возможность посмотреть номиналы паразитных конденсаторов и резисторов, а также подсветить их на схеме. Значения емкостей можно вывести сразу. В окне Schematic Editor выберем меню «Tools/Parasitics…», затем «Parasitics/Setup…». Рассмотрев содержимое открывшегося окна, убедимся, что в поле Extracted Cellview выставлены правильные данные и нажмем **OK**. Далее команда «Parasitics/Show Parasitics…» выведет на схему номиналы всех паразитных конденсаторов. С помощью меню «Parasitics…» можно также получить информацию о номиналах паразитных элементов между любыми двумя проводниками. Вывести номиналы резисторов можно только после проведения анализа схемы по постоянному току (DC-анализа).

- Теперь следует промоделировать схему компаратора вместе с паразитными элементами топологии. В конфигурации проекта заменим для компаратора вид schematic на только что созданный вид av_extracted и промоделируем АЦП. В список режимов анализа добавим также DC-анализ, чтобы по окончании расчета была возможность посмотреть номиналы паразитных резисторов. Если по результатам моделирования компаратор перестал работать, нужно проверить, не забыли ли сделать порты питания в проекте топологии.

- Убедившись, что изменение характеристик компараторов, вызванное учетом особенностей их топологической реализации, существенно не ухудшило параметров АЦП, можно двигаться дальше. В противном случае топологию придется переделывать.

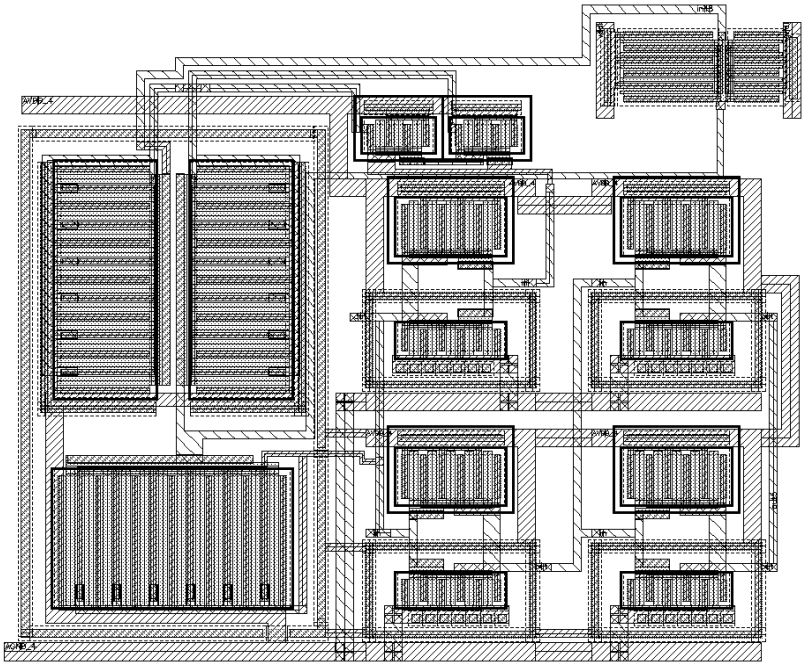


Рис. 48. Топология компаратора, разработанная вручную

Для цифрового блока нашего АЦП – шифратора temp_dc, нет необходимости вручную проектировать топологию. Она может быть создана автоматически с помощью специальных инструментов синтеза, например Cadence SoC Encounter. Соответствующий вид layout встраивается в базу данных Cadence и подлежит проверке с помощью средств Assura точно так же, как и нарисованные вручную аналоговые блоки. Топология шифратора, полученная таким способом, представлена на рис. 49.

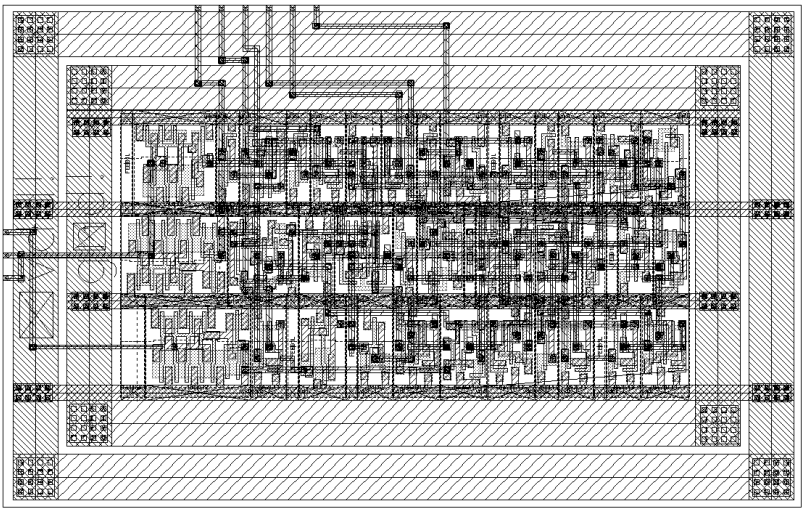


Рис. 49. Топология цифрового шифратора, созданная программой автоматического синтеза Cadence SoC Encounter

Некоторые полезные возможности редактора Layout XL.

- *Переименование межсоединений*. Когда элемент схемы импортируется в топологию, портам на топологии присваиваются имена, соответствующие именам цепей, подключенных к этим портам в схеме. Если затем схема изменится или если изменятся хотя бы названия цепей, редактор топологии не будет подсвечивать межсоединения, поскольку он ориентируется по именам. Чтобы исправить ситуацию, придется вручную переписать названия цепей, подключенных к портам элемента топологии. Такие свойства есть у каждого элемента, а отредактировать их можно с помощью меню «Connectivity/Propagate Nets…».

- *Создание охранных структур*. Часто требуется рисовать структуры, состоящие из набора слоев. Например, n^+ охранное кольцо обычно состоит из слоя NIMP (имплантация), DIFF (диффузия), MET1 (1-ый металл) и CONT (контакт). Такие структуры лег-

ко рисуются с помощью инструмента Multipart Path. Можно указать все слои, смещения между ними, отдельные настройки для квадратных элементов (контакты). Можно настроить один раз и сохранить, чтобы использовать в дальнейшем.

- *Создание охранных колец*. С помощью программы Layout XL можно быстро создавать охранные кольца на основе настроенных ранее структур Multipart Path. Для этого нужно выделить блок и выбрать в меню пункт «Create/Guard Ring».

- *Динамическая проверка соответствия технологическим нормам*. Для облегчения работы можно включить режим динамической проверки DRC-правил (меню «Options/DRD Edit»). В этом режиме ваши действия будут постоянно контролироваться, и в случае несоблюдения определенных норм, например, слишком близкого размещения элементов, соответствующая информация будет выведена на экран. Это полезная, но ограниченная функция, так как она отслеживает далеко не все правила. Кроме того, она не отменяет проверку верификатором Assura DRC. Этот вид проверки проводится обязательно во всех случаях.

Некоторые полезные опции верификатора Assura.

- По умолчанию размер выходного файла данных Assura ограничен величиной 2 Гбайт. В больших проектах этого объема зачастую не хватает, и процесс верификации завершается с ошибкой. Ситуацию можно исправить, воспользовавшись ключом (avParameter) ?diskList. Данная опция позволяет расширить базу данных до нескольких файлов по 2 Гбайт.

- Часто возникает необходимость «виртуального» замыкания цепи. Например, в аналого-цифровых системах аналоговые и цифровые общие (земляные) шины разводятся разными трассировочными проводниками и выводятся на разные контактные площадки. Эти шины в проекте имеют разное название (например, GNDA и GNDD). Однако из-за общего соединения через подложку Assura будет считать их одной цепью, что может вызвать ряд ошибок. Виртуально объединить их можно с помощью функции joinableNet. Например: joinableNet(cell(root) “gnda!” “gndd!”).

Выводы

По завершении рассмотренного этапа мы существенно приблизились к концу долгого пути проектирования нашего устройства. Представили и изобразили в виде топологического плана будущий кристалл. Также подготовили и проверили топологические маски основных его узлов – компаратора и шифратора. Результаты моделирования АЦП, блоки которого реализованы на уровне проекта топологии, приведены в табл. 13. Теперь предстоит выполнить заключительный этап работы. Он включает подготовку топологии верхнего уровня, окончательное моделирование всего кристалла и отправку базы данных на фабрику.

Таблица 13

Уровень абстракции моделей	SFDR, дБ	SNDR, дБ	ENOB, бит
Модель на поведенческом уровне	28	19,9	3,0
Модель на транзисторном уровне (типовые МОП транзисторы)	29	19,8	2,99
Модель с компараторами и шифратором на уровне топологии (типовые МОП транзисторы)	28	19,8	2,99

3.9. Проектирование топологии верхнего уровня

3.9.1. Введение

В этом разделе рассмотрим следующие вопросы: как топологию сложнофункционального блока (СФ блока), в данном случае нашего АЦП, превратить в топологию законченной микросхемы; как защитить будущую микросхему от повреждений из-за тиристорного эффекта (защелкивания) и электростатического разряда; как избежать потери точности и сбоев в работе схемы из-за распространения помех и шумов по подложке?

В чем разница между СФ блоком и готовой микросхемой? В общем случае СФ блок – законченное, аппаратно-программное, схемно-топологически верифицированное решение, представляющее собой продукт интеллектуальной собственности. Другими сло-

вами, это законченный строительный кирпичик, который может быть использован в дальнейшем. Если СФ блок проверялся не только в САПР, но также был изготовлен и протестирован, то его цена значительно возрастает. Многие компании занимаются разработкой СФ блоков наиболее востребованных устройств и типовых узлов. Их предлагают дизайн-центрам, не желающим тратить время на разработку и предпочитающим приобрести готовые решения.

Разработанный нами СФ блок может быть представлен на разных уровнях, включая законченную и верифицированную топологию устройства (рис. 50), которую можно использовать в различных проектах или на определенных условиях передать другим разработчикам.

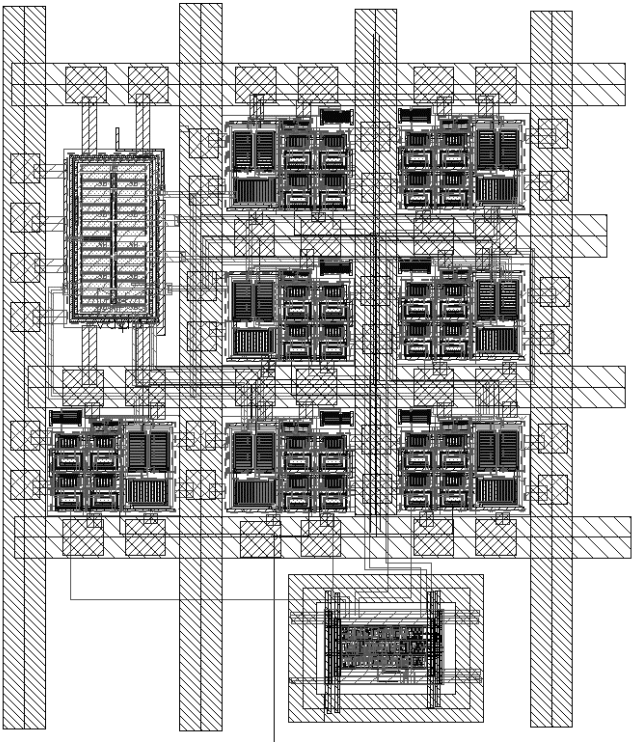


Рис. 50. Топология СФ блока аналого-цифрового преобразователя

Этот блок прошел проверки DRC и LVS, был промоделирован с включенными паразитными элементами топологии. Таким образом, его можно рассматривать как законченное техническое решение и продукт интеллектуальной собственности (СФ блок). Единственно, что с ним нельзя сделать – это отдать на изготовление без дополнительной подготовки.

Чтобы изготовить чип, помимо топологии блоков устройства необходимо создать топологию верхнего уровня, например кольцо контактных площадок для обеспечения связи с внешним миром. Далее рассмотрим несколько эффектов, которые надо учитывать при сборке топологии верхнего уровня, а также познакомимся с интерфейсами микросхемы.

3.9.2. Антенный эффект

Топологические ошибки, связанные с этим эффектом, легко обнаруживаются при DRC-верификации. Однако рассмотрим его, потому что обусловленные этим эффектом проблемы часто неожиданно возникают при подготовке топологии верхнего уровня.

В технологическом цикле производства микросхемы используется плазменное травление, которое требует нахождения подложки в области электрического поля высокой напряженности, ионизирующего плазму [23]. Во время травления поликремния электростатический заряд может накапливаться на поликремневом затворе. В результате напряжение может оказаться столь большим, что через подзатворный диэлектрик начнет протекать ток. Несмотря на то, что энергии может не хватить для пробоя, прочность оксида снизится пропорционально перенесенному через него заряду, отнесенному к общей площади диэлектрика. Накопленный областью поликремния заряд пропорционален ее площади. В результате присоединение маленького фрагмента подзатворного оксида к большой области поликремния может вызвать значительные повреждения тонкого оксида.

Этот явление называют *антенным эффектом*, потому что большая по площади область поликремния действует как антенна, накапливающая заряд, протекающий через уязвимый подзатворный диэлектрик. Повреждения, вызванные этим эффектом, происходят,

как правило, во время процесса имплантации при создании областей стока и истока МОП транзисторов.

Если DRC-верификатор выявил ошибки, связанные с антенным эффектом, исправить их можно двумя способами (рис. 51).

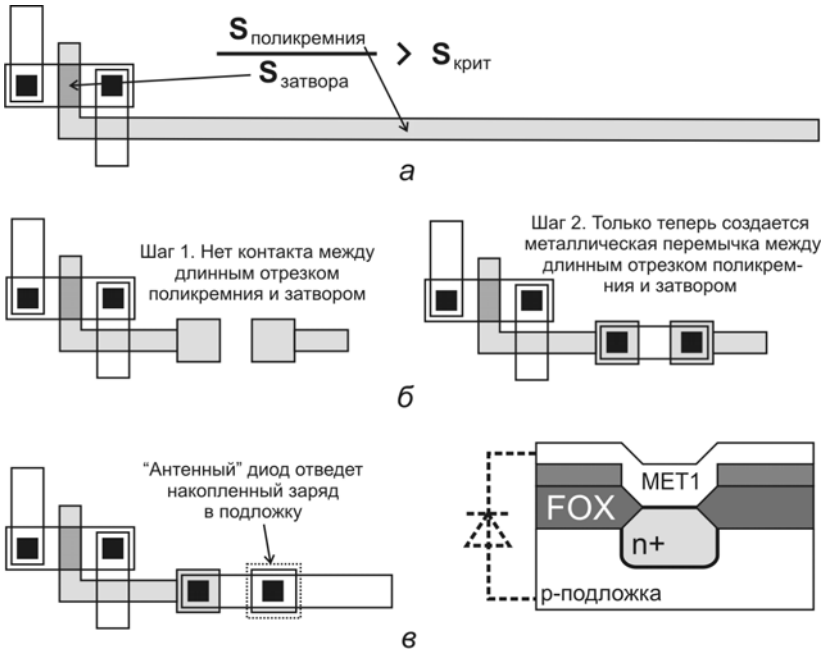


Рис. 51. Структура, которая может быть повреждена из-за «антенного» эффекта (*а*), и возможные методы защиты: создание металлической перемычки между затвором МОП транзистора и антенной (*б*), включение диода для отведения в подложку заряда, накопленного антенной (*в*)

В первом варианте «антенна» разбивается вблизи затвора короткой металлической перемычкой. Метод основан на том, что технологические операции выполняются последовательно: сначала создают слой поликремния, а потом металла. Поэтому при травлении поликремния образовавшаяся «антенна» будет слишком короткой,

чтобы повредить транзистор, а заряд, накопленный в металле во время следующей операции, будет также незначительным, поскольку перемычка – короткая. Второй вариант – использовать так называемый «антенный» диод, который отведет лишний заряд с длинной металлической линии в подложку.

3.9.3. Защита от защелкивания

Один из самых неприятных видов отказа ИМС связан с эффектом защелки или тиристорным эффектом. Две одинаковые микросхемы, испытываемые в одинаковых условиях, могут проявить себя абсолютно по-разному: одна будет работать, а вторая внезапно сгорит, подвергнувшись защелкиванию. Более того, микросхема может правильно функционировать сотни часов, а затем отказать.

Этот эффект связан с внезапным включением паразитного тиристора, образованного стандартными элементами типовой КМОП структуры (подложка, карман и области стока-истока). Защелкивание может быть спровоцировано резкими перепадами по шинам питания, в результате которых напряжение в части компонентов схемы поднимется выше общего напряжения питания или ниже «земли». Причины таких перепадов разные: включение и выключение питания, небольшой электростатический разряд на одной из ножек микросхемы, индуктивные наводки от расположенных рядом на плате мощных блоков. Защелкивание сопровождается резким увеличением сквозного тока, что при отсутствии ограничения тока на безопасном уровне может привести к перегреву и повреждению ИМС. Остановить этот процесс можно, лишь отключив питание микросхемы.

Наиболее высок риск возникновения тиристорного эффекта у интерфейсных элементов, подключаемых непосредственно к контактным площадкам микросхемы, потому что протекающие в них токи значительные, а размеры паразитных структур большие. Известно, что под влиянием переходного процесса напряжение на выходе обыкновенного инвертора в момент переключения может выходить за рамки номинальных уровней «земли» и питания. Это создает потенциальную угрозу защелкивания таких структур, особенно если инвертор используется в качестве выходного драйвера.

Эффект защелки трудно моделируется в САПР, поэтому принятие мер по его предотвращению полностью ложится на плечи разработчика топологии. Многие фабрики выпускают особые рекомендации для своих клиентов, соблюдая которые, можно многократно снизить риск защелкивания. В качестве примера приведем здесь рекомендации компании UMC для разработчиков микросхем с проектными нормами 0,18 мкм [29].

Рекомендации для интерфейсных элементов:

- исток каждого *p*-канального транзистора необходимо напрямую, используя единственную линию, без перехода в другие слои металла подключать к шине питания VDD;
- исток каждого *n*-канального транзистора необходимо напрямую, используя единственную линию металла, подключать к шине «земли» VSS;
- стоки *n*- и *p*-канальных выходных транзисторов необходимо напрямую подключать к контактным площадкам (КП), используя единственную линию металла; категорически запрещены переходы в другие слои – так называемые «подныры»;
- каждый *p*-канальный транзистор, подключаемый к контактной площадке, необходимо окружить *n*+ диффузионным кольцом, непосредственно соединенным с шиной питания VDD;
- каждый *p*-канальный транзистор, подключаемый к контактной площадке, необходимо окружить с внешней стороны по периметру кармана дополнительным *p*+ охранным кольцом, напрямую присоединенным к шине VSS (рис. 52);
- каждый *n*-канальный транзистор, подключаемый к контактной площадке, необходимо окружить *p*+ диффузионным кольцом, непосредственно соединенным с шиной VSS;
- каждый *n*-канальный транзистор, подключаемый к контактной площадке, необходимо окружить по периметру дополнительным *n*+ охранным кольцом, напрямую присоединенным к шине питания VDD (см. рис. 52);
- минимальная ширина всех охранных колец должна составлять 3 мкм;
- расстояние от края *n*-канального транзистора до границы *p*-канального транзистора в интерфейсной схеме должно быть не менее 20 мкм.

Рекомендации для внутренних блоков:

- внутренние и интерфейсные блоки должны быть разделены двойным охранным кольцом *n*- и *p*-типа шириной более 3 мкм;
- в кармане *n*-типа расстояние между границей *p*-МОП структуры и ближайшим контактом к карману должно быть не более 20 мкм;
- расстояние между границей *n*-МОП структуры и ближайшим контактом к подложке также не должно быть более 20 мкм;
- расстояние между интерфейсными транзисторами на периферии кристалла и ближайшими к ним транзисторами внутренней схемы должно быть не менее 80 мкм (см. рис. 52).

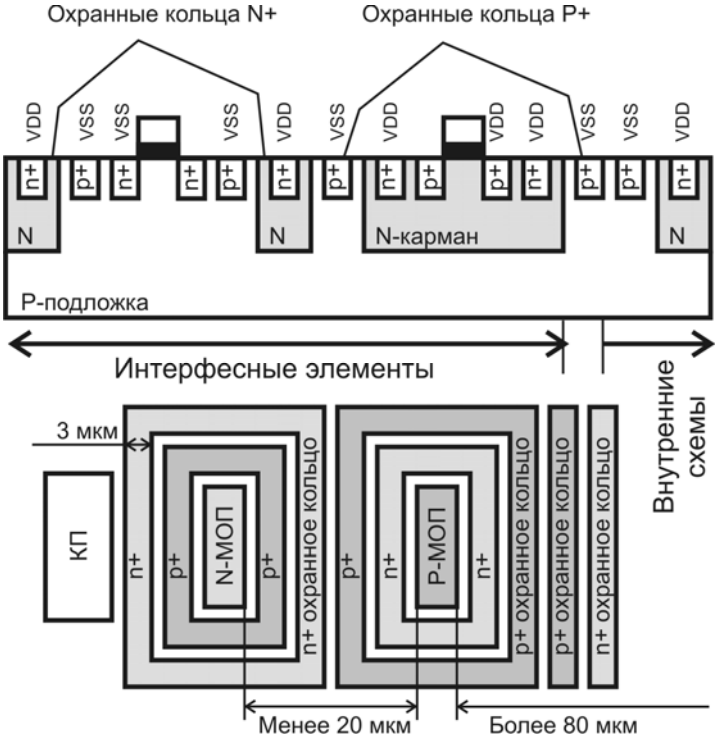


Рис. 52. Рекомендации по проектированию топологии ИМС с повышенной устойчивостью к эффекту защелкивания

3.9.4. Взаимное влияние аналоговых и цифровых блоков

Интеграция на одном кристалле блоков, выполняющих аналоговые и цифровые функции, рождает ряд дополнительных проблем, связанных с взаимным влиянием этих блоков. Оно возникает благодаря распространению нежелательных помех от цифровых элементов к чувствительным аналоговым узлам. В основном цифровые блоки генерируют помехи по шинам питания и по подложке, внося высокочастотный шум в аналоговые компоненты. Термином «шум» обозначаем сигнал помехи, обусловленной переключениями совокупности элементов цифровых блоков и имеющий шумоподобный спектр. В этом разделе рассмотрим меры, препятствующие распространению этого вида помех, уменьшению наводок на аналоговые блоки и улучшающие характеристики аналого-цифровых систем.

Распространение шумов по шинам питания. Объединение шин аналоговой и цифровой «земли» приводит к возникновению помехи в виде паразитных осцилляций на аналоговом выходе [30]. Они имеют вид затухающих колебаний, максимальная амплитуда которых пропорциональна эффективной величине индуктивности общей шины, включающей индуктивность шины «земли» на кристалле и индуктивность проводника, соединяющего контакты «земли» на кристалле и на корпусе ИМС. Для соединения контактных площадок чипа и корпуса обычно применяют золотую или алюминиевую проволоку длиной от 5 до 10 мм. Индуктивность такой проволоки составляет приблизительно 1 нГн/мм. На высоких частотах и при быстрых переходных процессах индуктивный импеданс контактного проводника может дать весьма существенный вклад в общее сопротивление шины. Например, при резких бросках тока, потребляемого по цепи питания, возникает значительное падение напряжения между выводом корпуса микросхемы и контактной площадкой чипа. Снижение напряжений питания блоков ИМС может достигать 500 мВ, что более всего влияет на характеристики чувствительных к изменениям уровней питающих напряжений аналоговых блоков.

Лучшее решение проблемы – отказаться от общих шин как «земли», так и питания и сделать их раздельными (рис. 53, *а*). Цена такого решения – необходимость в дополнительных выводах кор-

пуса. Число выводов корпусов современных БИС непрерывно растет, в том числе и по этой причине. В микросхемах, содержащих многочисленные цифровые и аналоговые узлы, число выводов корпуса часто достигает нескольких сотен, из которых значительная часть является выводами шин «земли" и питания.

Контакты чипа и корпуса являются весьма ценным ресурсом, во многом определяющим стоимость ИМС. Поэтому, если возможность разделения шин «земли» и питания различных аналоговых и цифровых блоков отсутствует, используют два других менее эффективных способа (рис. 53, *б* и *в*). В обоих вариантах создается фильтр между аналоговой и цифровой шиной, уменьшающий их взаимное влияние. Вариант, показанный на рис. 53, *б*, лучше, чем на рис. 53, *в*, так как в последнем варианте внешняя фильтрующая емкость, а также емкость вывода изолированы от шин питания индуктивностью проводника, соединяющего выводы чипа и корпуса ИМС. Вместе с тем, в варианте на рис. 53, *в* меньше число контактных площадок на кристалле, что позволяет сократить его площадь.

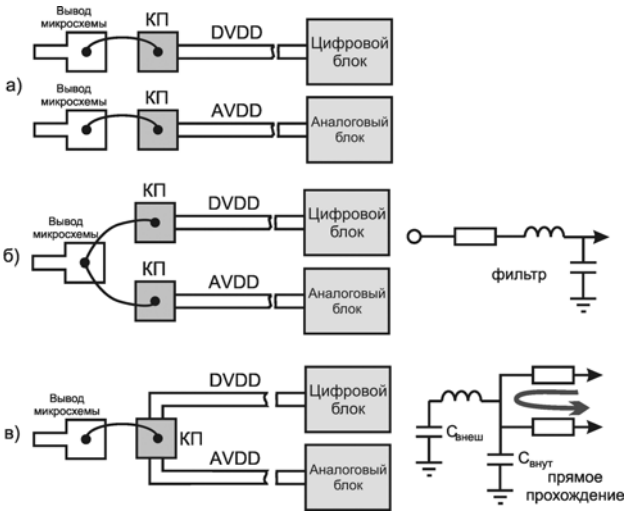


Рис. 53. Способы, препятствующие распространению шумов из цифровых блоков в аналоговые по шинам питания: раздельные контакты чипа и корпуса (*а*), раздельные контакты чипа и общий вывод корпуса (*б*), общий контакт чипа и общий вывод корпуса (*в*)

Распространение шумов по подложке. Общая подложка способствует передаче шумов от цифровых блоков к аналоговым. На рис. 54 показаны два типа кристаллов: с эпитаксиальным слоем на низкоомной подложке (см. *а*) и без эпитаксии (см. *б*). Эпитаксиальный слой обычно слабо легирован и обладает относительно высоким сопротивлением, подложка же, наоборот, имеет небольшое сопротивление. Как следствие, движение тока между элементами, расположенными в эпитаксиальном слое, происходит по пути наименьшего сопротивления – в вертикальном направлении (см. рис. 54, *а*). Поскольку подложка имеет малое сопротивление, то расстояние между источником и приемником шума практически не имеет значения. Естественно, при чрезмерном сближении объектов начнут играть роль горизонтальные составляющие тока, проходящие по эпитаксиальному слою. Существует простое правило: если расстояние между двумя диффузионными областями в четыре и более раз меньше, чем толщина эпитаксиального слоя, то горизонтальные токи следует учитывать [31].

В случае, показанном на рис. 54, *б*, подложка равномерно легирована и характеризуется высоким сопротивлением. Значит, ток из левой области $p+$ потечет в правую область по всем возможным направлениям. В этом случае расстояние между источником и приемником шума становится важным – чем больше расстояние, тем больше сопротивление.

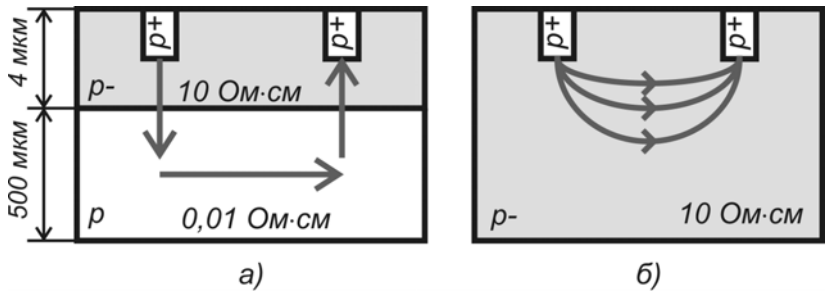


Рис. 54. Пути распространения шумов и помех по кристаллу с эпитаксиальным слоем (*а*) и без эпитаксии (*б*)

Лучшим способом борьбы с помехами этого вида, практически исключающим распространение шума по подложке, является использование технологии «кремний на изоляторе». Однако из-за высокой стоимости она применяется, в основном, для создания специальных изделий. Два других наиболее часто используемых метода экранирования, препятствующих распространению шума по подложке, показаны на рис. 55. Согласно первому (рис. 55, *а*) в *p*-подложке создается *n*-карман, который подключается к наивысшему потенциалу схемы. Возникающая обедненная область играет роль своего рода барьера, препятствуя протеканию шумового тока по короткому пути. Соответственно растет сопротивление, и помеха ослабляется. Второй способ – экранирование поверхностных и горизонтальных токов с помощью охранных колец – высоколегированных диффузионных областей, окружающих чувствительные аналоговые элементы (рис. 55, *б*). Охранные кольца подключаются к низшему потенциалу схемы. Отметим, что нежелательно непосредственно соединять охранные кольца с подложкой на кристалле, так как помеха может пройти через охранное кольцо в подложку рядом с чувствительным элементом. Контактов к охранному кольцу должно быть как можно больше.

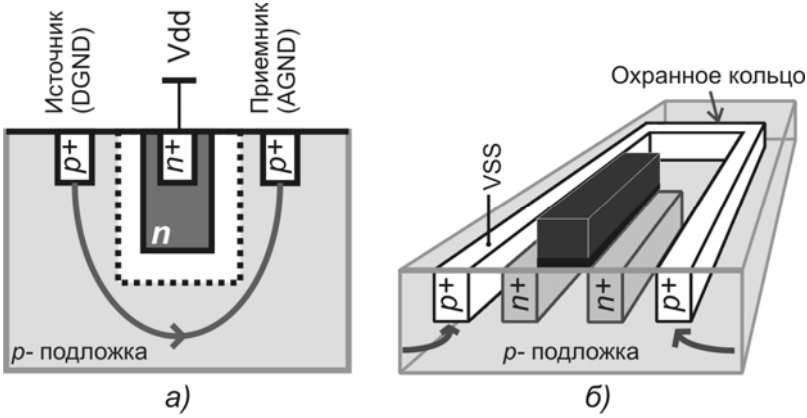


Рис. 55. Способы экранирования распространения шумов по подложке: с использованием барьера на основе кармана *n*-типа (*а*), с использованием охранного кольца (*б*)

Еще один метод экранирования, обеспечивающий большую степень защиты от взаимного влияния элементов по подложке, чем два предыдущих способа, показан на рис. 56. К сожалению, он доступен, только если используемая технология позволяет делать глубокие карманы. Такой *n*-карман становится своего рода локальной подложкой для внутренней схемы. Он подключается к изолированному потенциалу, а внутри глубокого кармана создаются более мелкие карманы *p*-типа.

Итак, если позволяет технология, конечно, используем глубокие карманы. Кроме того, два первых способа хорошо сочетаются и в совокупности дают неплохой результат, поэтому не стоит забывать и о них. Общая рекомендация состоит в том, что аналоговые и цифровые блоки желательно размещать на максимально возможных расстояниях друг от друга и разделять их широкими защитными областями. Следует также отметить, что уровень шума в подложке пропорционален напряжению питания [32]. По этой причине выходные драйверы шумят больше внутренних схем, так как обычно питаются от источников с более высоким напряжением, чем ядро ИМС.

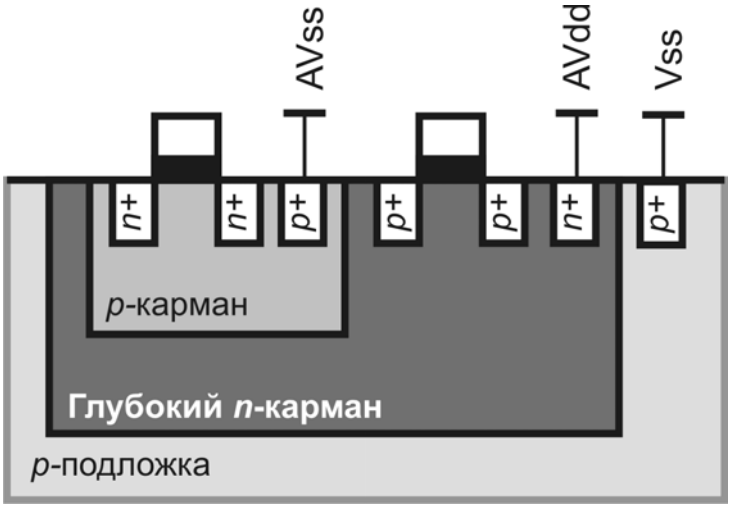


Рис. 56. Способ экранирования шумов в технологии с глубокими карманами

Оценим минимальное количество выводов, необходимое для качественной изоляции от шумов по питанию и подложке:

1). аналоговая «земля»;
2). аналоговое питание;
3). цифровая «земля»;
4). цифровое питание;
5). охранные кольца *n*-типа;
6). охранные кольца *p*-типа;
7). подложка.

Итого - семь выводов. Такова цена качественной изоляции.

3.9.5. Защита от электростатического разряда

Электростатический разряд (ЭСР, ESD) приводит к серьезным последствиям, включая пробой и повреждение диэлектрика, а также лавинный пробой *p-n* перехода. При особенно сильном ЭСР возможно даже испарение металлизации и разрушение кремния [23].

Самый распространенный источник статического электричества – человеческое тело. При ходьбе по поверхности, обладающей диэлектрическими свойствами, на теле может накопиться заряд, создающий электростатический потенциал относительно окружающих предметов величиной несколько киловольт, иногда даже свыше 20 кВ. Поэтому если коснуться выводов микросхемы незаземленной рукой, может возникнуть ЭСР соответствующей амплитуды. Несмотря на то, что разрядный импульс – короткий (порядка 200 нс), этого вполне достаточно для необратимых повреждений микросхемы.

Проблема защиты от статического электричества особенно актуальна для современных КМОП микросхем, в которых толщина подзатворного диэлектрика составляет десятки ангстрем. Этой проблеме посвящено множество работ, например [33, 34], где подробно как с теоретической, так и с практической точки зрения, рассматриваются способы ее решения. Моделировать защиту от ЭСР очень сложно. Нужны долговременные исследования, выпуск множества тестовых образцов. Поэтому разработчики микросхем редко занимаются таким моделированием. Обычно эту работу выполняют фабрики.

Они включают в состав пакетов своих проектных библиотек набор входных и выходных элементов, защищенных от статического электричества. Часто они представлены как библиотеки входных и выходных контактов, а фактически это наборы стандартных ячеек, подключенных к контактным площадкам. Такие стандартные ячейки можно использовать в своих проектах.

Назначение защиты от ЭСР – отвести избыточный заряд от уязвимых внутренних схем. При этом важно обеспечить равномерное растекание избыточного тока, своевременное включение защитного элемента и быстрый отвод тока с наименьшим рассеиванием тепла [9]. В качестве защитных элементов используются обратно смещенные диоды на основе перехода сток-подложка МОП транзисторов. Транзисторы имеют широкую диффузионную область стока со специальным салицидным (salicide, self-align silicide) покрытием для формирования балластного резистора. С целью предотвращения защелкивания используются охранные кольца. Стандартная защита выдерживает напряжение 2000 В, если выполнять тестирование в соответствии с моделью человеческого тела [9] и напряжение 200 В, если для тестирования использовать так называемую машинную модель.

На рис. 57 показан пример организации защиты от статического электричества на базе стандартных ячеек окружения внешних выводов ИМС (pad based ESD protection). ЭСР отводится по двум периферийным шинам питания VDDe и VSSe. Каждая входная и выходная ячейка помимо драйвера содержит элементы защиты от ЭСР, представляющие собой *n*- и *p*-канальные транзисторы особой топологии. Защита ряда аналоговых блоков может отличаться от защиты других блоков типами используемых компонентов из-за необходимости компромисса между степенью защиты и дополнительными паразитными эффектами, вносимыми этими компонентами. Выводы питания и «земли» также защищаются от воздействия статического электричества. Дополнительные элементы защиты часто встраиваются даже в каждый из углов кристалла микросхемы.

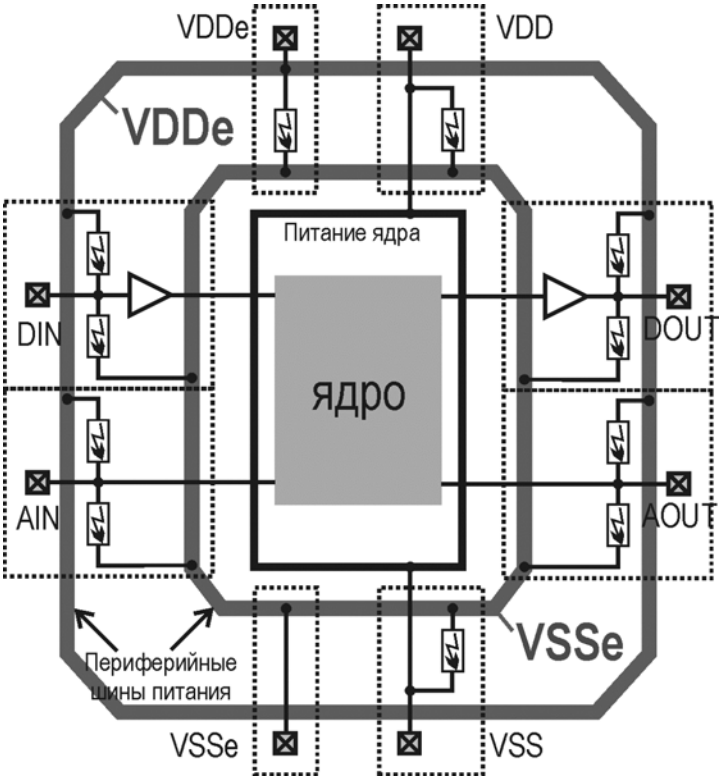


Рис. 57. Защита ИМС от электростатического разряда

Ряд замечаний, которые необходимо учитывать, используя стандартные элементы защиты от ЭСР.

- Цифровые ячейки имеют в своем составе внутренние буферы, содержащие необходимую защиту от защелкивания. Аналоговые ячейки не имеют буферов. Их входы и выходы соединены непосредственно с внутренними блоками, поэтому защита от защелкивания ложится на плечи разработчика.
- Не рекомендуется разбивать периферийные шины на отрезки. Если это все же необходимо, например для разводки аналогового и цифрового питания, следует пользоваться стандартными специальными ячейками типа cut-cells. Но и в этом случае общую

«землю» периферийного кольца защиты от ЭСР прерывать не стоит, чтобы заряд с любого участка шины мог беспрепятственно стекать на соответствующую ножку микросхемы.

- Механические воздействия на кристалл при резке пластины могут повредить металлические дорожки периферийных шин, расположенные поблизости от углов кристалла. Кроме того, по кольцам периферийных шин могут протекать значительные токи. Поэтому у этих колец обрезают углы под наклоном 45 градусов, а в углах кристалла не размещают контактные площадки. Однако в них можно расположить дополнительные элементы защиты от статического электричества. Соответствующие стандартные ячейки обычно содержатся в библиотеке входных и выходных элементов.
- Не рекомендуется проводить пути оттока заряда статического электричества через внутренние блоки. Периферийные шины питания и цепи защиты от ЭСР не должны объединяться с внутренними шинами питания, а должны иметь отдельные контактные площадки.

3.9.6. Контактные площадки

Любая микросхема должна иметь интерфейс для связи с внешним миром. Интегральные схемы подключаются к другим электронным компонентам через выводы корпуса. Вплоть до разделения выводов на финальной стадии изготовления ИМС они объединены монтажной рамкой, как показано на рис. 58. Внутри корпуса выводы подводятся к контактным площадкам, расположенным по периметру зоны расположения чипа. При монтаже кристалла его контактные площадки соединяются отрезками золотой или алюминиевой проволоки с выводами корпуса методом сварки или термокомпрессии.

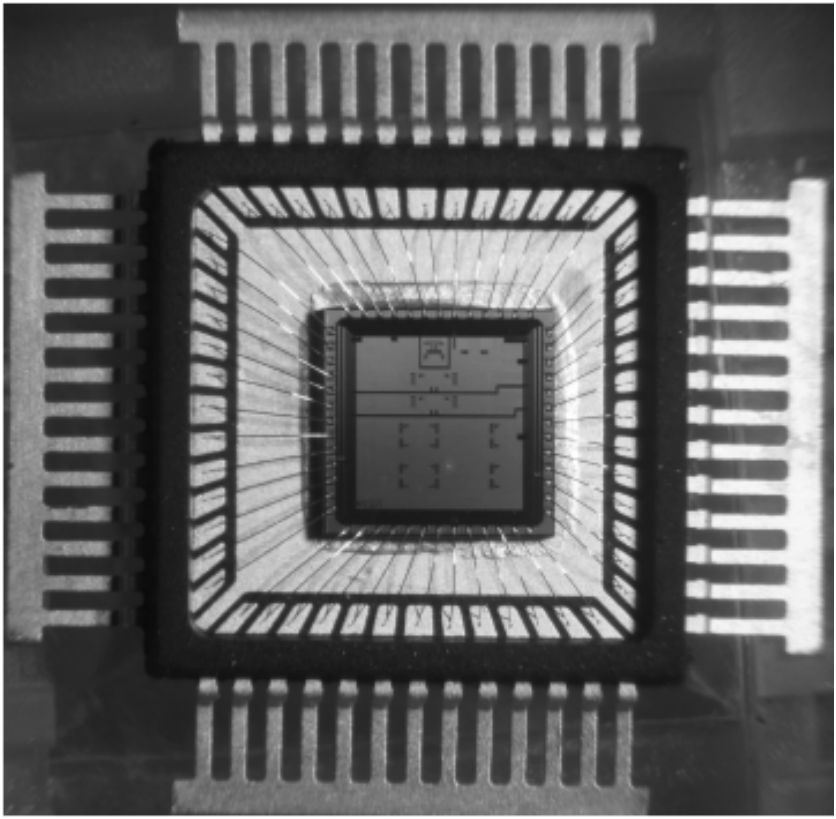


Рис. 58. Расположение кристалла в корпусе ИМС

Контактная площадка (КП) топологически представляет собой совокупность металлических площадок, расположенных друг над другом, наподобие слоеного пирога (рис. 59). Сверху донизу слои этой структуры пронизывают контактные окна, расположенные в шахматном порядке. Понятно, что чем больше таких окон, тем меньше паразитное сопротивление площадки.

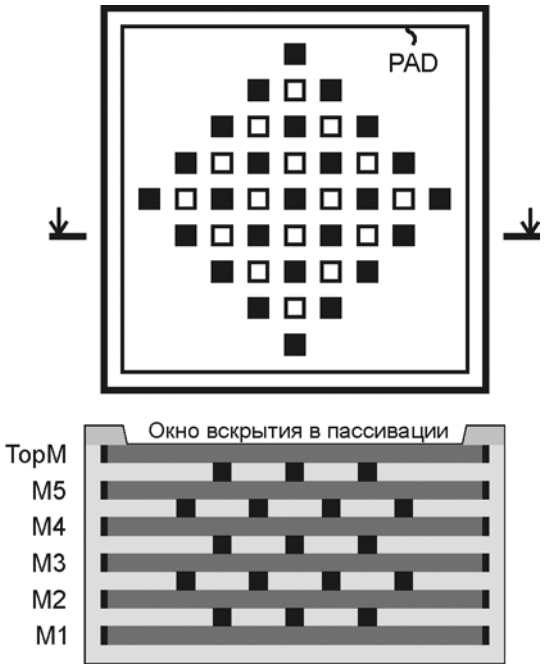


Рис. 59. Структура контактной площадки ИМС

Для защиты от влаги и повреждения кристалл микросхемы покрывают специальным защитным слоем. Эта операция называется пассивацией. Типовым защитным покрытием в технологическом процессе с проектными нормами 0,18 мкм служит полиимидная пленка толщиной 1,5 – 2,0 мкм. Пассивация не только защищает от влаги, но является также хорошим изолятором. Следовательно, в месте контакта площадки с проволокой необходимо сделать окно вскрытия. В топологии слой вскрытия обычно обозначается PAD. Если по ошибке этот слой отсутствует, то подключиться к микросхеме будет невозможно. За этим надо внимательно следить при подготовке топологии верхнего уровня.

Размер контактной площадки, а точнее окна вскрытия, в стандартных ячейках обычно составляет 60 х 60 мкм. Размер окна при необходимости может быть увеличен, например, если этого требует используемая технология сборки в корпус. Кстати, окна вскры-

тия можно делать не только для контактных площадок. Многие технологии позволяют вскрывать небольшие окна размерами до 6 х 6 мкм над специальными областями металла (probe pad), предназначенными для тестирования ИМС. Через эти окна с помощью специальных игл можно контролировать потенциалы и форму сигналов в нужных точках.

Стандартная ячейка КП, поставляемая в составе пакета проектных библиотек, – не только собственно контакт, но и дополнительный набор элементов, а именно:

- схема защиты от статического электричества;
- интерфейс с внутренней схемой кристалла;
- блоки, реализующие дополнительные функции, например буфер с тремя состояниями.

В итоге топологический размер ячейки разрастается и может занять значительную площадь на кристалле. В связи с этим разработчики проектных библиотек оптимизируют стандартные ячейки КП по площади, приводя их к двум основным вариантам. Первый вариант предназначен для кристаллов, минимальный размер которых ограничивается контактными площадками (pad-limited). Они имеют большое число выводов, но относительно небольшое по площади ядро, что характерно для цифровых БИС. В этом случае КП вместе с их интерфейсами располагаются настолько близко друг к другу, насколько это позволяют нормы. В результате ячейки вытягиваются, занимая значительную площадь внутри кристалла (рис. 60, *а*). Их называют pad-limited ячейками. Мы также будем использовать это название за отсутствием иного адекватного по значению краткого термина.

Второй тип ячеек КП предназначен для кристаллов с небольшим числом выводов, но большим ядром, площадь которого и ограничивает его минимальный размер (core-limited). Если периметра ядра достаточно для размещения всех КП, необходимо использовать этот тип ячеек (рис. 62, *б*). Они имеют меньшую высоту, что позволяет оптимизировать площадь микросхемы с большим ядром. По причине изложенной выше мы будем их назвать core-limited ячейками.

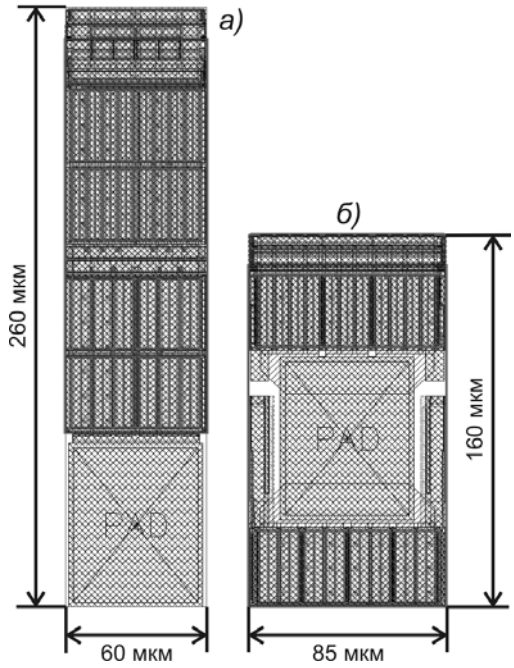


Рис. 60. Разновидности стандартных ячеек, связанных с внешними контактными площадками: ячейка pad-limited, используемая в проектах, где минимальный размер чипа ограничен площадью периферийных контактов, (*а*) и ячейка core-limited, применяемая в проектах, где размер чипа определяется площадью ядра, (*б*)

Основной недостаток применения core-limited ячеек состоит в том, что для снижения их высоты уменьшают ширину периферийных шин, к которым относятся не только цепи для отвода электростатического заряда, но и шины питания интерфейсных буферов. Вследствие недостаточной ширины трасс допустимый ток ограничивается, а это значит, что мощные буферы с высокой нагрузочной способностью не могут быть использованы совместно с такими ячейками. В связи с этим core-limited ячейки рекомендуется применять только в небольших чипах для минимизации площади, зани-

маемой периферийной частью. В больших микросхемах их использовать не следует из-за малой величины тока, пропускаемого периферийными шинами. По этой причине ограничено также число одновременно переключающихся выходных драйверов. Кроме того, такие ячейки обладают меньшей стойкостью к статическому электричеству.

Отметим, что напряжение питания периферийных блоков (драйверов, элементов защиты от ЭСР и др.) обычно превышает напряжение питания ядра. Типовыми для современных микросхем являются уровни напряжений от 0,8 до 1,8 В для питания ядра и 3,3 В, иногда 5 В для питания периферийных блоков. Поэтому отдельная периферийная шина, обеспечивающая отвод статического заряда, также имеет повышенное напряжение, соответственно 3,3 или 5 В. Эта же шина обычно служит для подачи питания на выходные драйверы.

3.9.7. Практическая работа в Cadence. Создание топологии верхнего уровня

- Откроем схему myadc.schematic. Выберем пункт «Tools/Design Synthesis/Layout XL» окна Schematic Editing. Создадим новый вид layout. В программе Virtuoso XL Layout Editing, используя импорт компонентов из схемы и подсветку межсоединений, нарисуем топологию СФ блока АЦП, показанную на рис. 50.
- Оценим занимаемую АЦП площадь. Это будет площадь ядра ИМС. Учтем также площадь 12-ти внешних контактных площадок. Необходимая длина периметра периферийной части микросхемы (контактные площадки, элементы защиты от ЭСР и шины питания) значительно превышает периметр ядра. Следовательно, используем стандартные pad-limited ячейки из библиотеки контактных площадок. Соответствующая библиотека стандартных ячеек КП должна быть в пакете проектных библиотек.
- Подключим питание. Одна пара шин «земли» и питания (GND, VDD) резервируется для отвода заряда при возникновении ЭСР и для подачи питания 3,3 В на выходные драйверы. Еще три линии проводников (две шины GND и одна VDD) служат для подачи питания 1,8 В на ядро микросхемы. Стандартные ячейки подключения питания устроены так, что контактные площадки уже

соединены с соответствующими периферийными шинами, поэтому нам остается лишь расположить ячейки в нужных местах чипа и замкнуть кольцо. Питание будет автоматически подано на него.

- Подключим аналоговые входы. Для этого понадобятся типовые ячейки контактных площадок, предназначенные для аналоговых входов и выходов. Используем их для создания выводов CLK, INP, VREF, INN.
- Создадим контакты для цифровых выходов D0, D1, D2. С этой целью используем стандартные ячейки контактных площадок, содержащие цифровые драйверы с напряжением питания 3,3 В и простой буферизацией (без третьего состояния). Транслятор логических уровней, который применяется при передаче сигналов от ядра с напряжением питания 1,8 В к периферийным блокам с напряжением питания 3,3 В, входит в состав стандартной ячейки контактной площадки.
- Соединим использованные ячейки контактных площадок. Для этого применим стандартные элементы стыковки периферийных ячеек. Среди них есть специальные ячейки, представляющие собой фрагменты шин, которые можно вплотную состыковать с ячейками контактных площадок так, что не будет разрывов проводников. Среди них имеются также угловые элементы, содержащие дополнительную защиту от статического электричества.
- Используя необходимые стандартные ячейки, соберем периферийное кольцо (pad ring) вокруг ядра АЦП как показано на рис. 61. Затем подведем к контактным площадкам соответствующие выводы ядра, а питание и «землю» подключим к кольцу в нескольких местах, чтоб не было «просадок» питающих напряжений. Топология готова.
- Проведем верификацию DRC. Убедимся, что ошибок нет. Если теперь выполнить верификацию LVS, то результат будет отрицательный – появятся сотни ошибок. Это связано с тем, что элементы периферийного кольца (а там много и транзисторов, и резисторов) есть в топологии, но отсутствуют в схеме. Возникает задача, обратная той, что является привычной для нас, а именно – создание схемы по топологии. Решается она просто, поскольку стандартные ячейки контактных площадок имеют не только топологическое, но и схемное представление. Нужно лишь собрать отдель-

ный блок, скажем pad_ring, и правильно соединить эти ячейки вместе. Оставляем эту задачу для самостоятельной проработки.

- Когда верификация LVS пройдет без ошибок, сделаем экстракцию всего АЦП и промоделируем полученную схему в нашем окружении. Может возникнуть ситуация, что вычислительной мощности используемых компьютеров уже не хватит на такой расчет. Это вполне вероятно, потому что периферийное кольцо содержит очень много компонентов. В частности, одни только контактные площадки могут иметь миллионы контактных окон! Проблема решается разбиением задачи на две подзадачи. Из топологии удаляются периферийные контактные площадки. Внутренние контакты топологии (пины) остаются. АЦП моделируется без миллионов дополнительных резисторов и емкостей. Полностью периферийное кольцо как результат экстракции моделируется в составе АЦП, который может иметь вид схемы или модели.

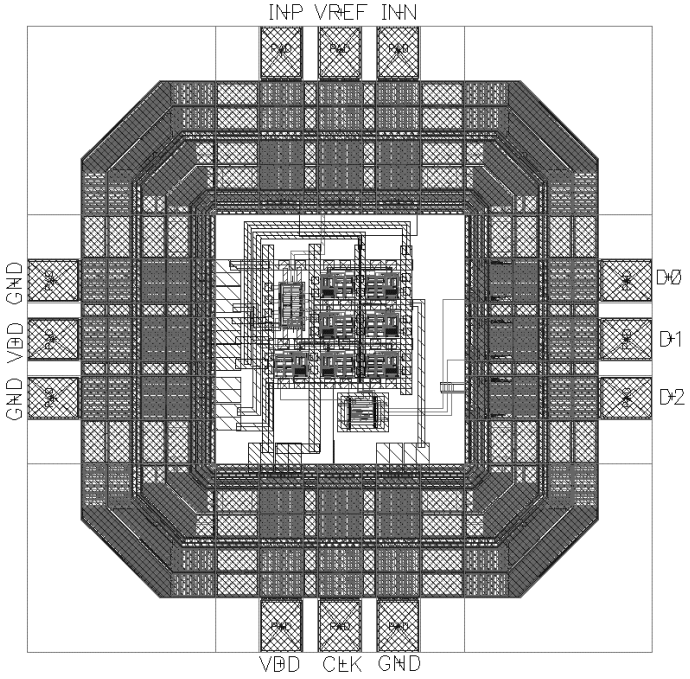


Рис. 61. Окончательный вид топологии АЦП

Выводы

Итак, долгий путь разработки АЦП подходит к концу. Мы познакомились с рядом факторов, которые оказывают существенное влияние на функционирование микросхем и которым следует уделять особое внимание при создании топологии ИМС: эффект защелкивания, антенный эффект, распространение шумов по подложке и шинам питания. Топологическое проектирование всего кристалла завершено. Проведена экстракция из топологии полной схемы чипа с учетом всех особенностей его технологической реализации и с учетом реального влияния всех паразитных элементов, которые возможно учесть, используя модели из пакета проектных библиотек. Выполнено моделирование схемы, полученной в результате экстракции из топологии чипа. Окончательные результаты моделирования кристалла совместно с данными, полученными на предыдущих этапах, приведены в табл. 14. Они значительно превышают требования ТЗ, что дает нам право ожидать хороших результатов испытаний опытных образцов.

Таблица 14

Уровень абстракции моделей	SFDR, дБ	SNDR дБ	ENOB бит
Модель на поведенческом уровне	28	19,9	3,0
Модель на транзисторном уровне (типовые МОП транзисторы)	29	19,8	2,99
Модель с компараторами и шифратором на уровне топологии (типовые МОП транзисторы)	28	19,8	2,99
Модель с учетом особенностей технологической реализации – экстракция из топологии чипа	26	18,4	2,68
Техническое задание	22	16	2,4

Работа почти завершена. Но перед изготовлением и тестированием микросхемы нам осталось сделать еще один небольшой, но очень важный шаг – экспорт базы данных и отправка топологии разработанного чипа на фабрику. Об этом поговорим в следующем, заключительном разделе.

3.10. Экспорт топологии для производителя ИМС

3.10.1. Особенности производства малых партий микросхем

Закончив работу над проектом, осталось выяснить, как организовать его передачу на микроэлектронную фабрику. Но прежде несколько слов о самой фабрике. Современные технологии производства настолько дороги, что изготовление одного комплекта фотошаблонов может стоить заказчику до 10 млн дол. Такие цены делают изготовление тестовых образцов и продукцию небольшого объема просто нерентабельным, поскольку начальные затраты часто не могут покрыться объемом производства.

Решением этой проблемы известным с 80-х годов прошлого века является особая форма организации производства и обслуживания разработчиков ИМС, которая основана на регулярном запуске на изготовление так называемых многопроектных пластин (Multi-Project Wafer, MPW, МПП). Суть этой формы производства и сервиса заключается в следующем: компания-организатор собирает заказы от различных фирм, дизайн-центров, отдельных групп разработчиков, студентов, выполняющих учебные работы, и компонует из них единый проект, на основе которого разрабатывается один комплект фотошаблонов. Затраты при этом делятся на всех участников пропорционально доли площади их кристаллов в суммарной занимаемой площади на пластине [35].

Организатором такого бизнеса может быть как сама фабрика-производитель МПП, так и специализированные компании и организации, например MOSIS или Europractice. Стоимость производства для каждого из участников таких коллективных проектов многократно снижается. Так цена квадратного миллиметра чипа, реализуемого по КМОП технологии с проектными нормами 0,18 мкм, хотя и колеблется в зависимости от производителя и опций процесса, но в среднем оказывается порядка 1000 евро.

Главными недостатками такой формы производства микросхем являются наличие жестко установленных сроков запуска пластин на изготовление и ограниченный объем выпускаемых образцов. Кроме того, есть ограничения по максимальной и минимальной площади кристаллов, а также вероятность не попасть в удобный по срокам запуск из-за большого конкурса среди желающих.

В целом практика участия в многопроектных запусках опытных образцов микросхем на ранних стадиях работы давно вошла в арсенал производителей электроники по всему миру. Это позволяет достичь существенного сокращения затрат и сроков выхода на рынок, одновременно накладывая отпечаток на стратегию разработки. Приведем несколько особенностей стратегии разработки микросхем с ориентацией на участие в многопроектном запуске в производство опытных образцов [36].

- При первом запуске не нужно изготавливать большое число образцов и необязательно все кристаллы монтировать в корпус, поскольку они предназначены только для тестирования. В дальнейшем исправление ошибок или просто улучшение характеристик ИМС может быть достигнуто простой заменой одной или нескольких масок металлизации без изменения остальных слоев. Такой повторный запуск (respin) оказывается существенно дешевле.
- Сначала необходимо сконцентрироваться только на достижении принципиальной работоспособности устройства, а не на достижении высокого уровня качественных показателей, максимизации выхода годных, минимизации площади и других аспектах. Даже самые крупные и опытные компании на сегодняшний день исходят из того, что любой проект требует хотя бы одного повторного запуска.
- На подготовке первого запуска создают дополнительные элементы, предназначенные для последующего тестирования схемы. Это могут быть специальные тестовые блоки, контакты под иголки зондов и др. На этом этапе возможна замена части внутренних блоков их внешними аналогами. Такие компоненты существенно облегчают тестирование и легко удаляются при последующих запусках.
- Если цели первого этапа достигнуты, то при последующих запусках можно не ограничивать количество изготовленных микросхем минимальной партией. Основную часть стоимости запуска составляет изготовление масок, поэтому заказ одного дополнительного чипа не добавит к цене больше 1 %.

Допустим, место на МПП зарезервировано. Крайний срок приема проектов, установленный организатором, обычно учитывает время, отведенное под входной контроль, на котором осуществля-

ется проверка соблюдения правил DRC. В случае возникновения ошибок клиенту отправляется соответствующее сообщение и дается время на их исправление. Первая проверка – бесплатная, за все последующие берется плата. Если клиент опаздывает к требуемому сроку, то иногда производитель идет ему навстречу и может принять проект даже в течение периода входного контроля. Однако в этом случае контроль проводиться не будет, и вся ответственность ложится на заказчика. Аналогичное положение возникает, если входной контроль выявил ошибки, а они исправлены не были. По окончании периода входного контроля начинается процесс компоновки кристаллов и подготовки единого комплекта фотошаблонов. После этого стартует технологический процесс изготовления микросхем, который длится обычно от двух до четырех месяцев.

Общий план подготовки проекта к запуску следующий:

- создание топологического файла в формате GDS-2 и его архивирование;
- проверка файла, позволяющая исключить появление ошибок при экспорте;
- заполнение необходимых форм и документов (обычно на WEB-сервере организатора);
- передача файла производителю (обычно закачка на закрытый FTP-сервер организатора).

3.10.2. Практическая работа в Cadence. Архивация и экспорт проекта

Подготовим проект к отправке на фабрику.

- Прежде всего, необходимо собрать весь проект в одной библиотеке, чтобы упорядочить базу данных и облегчить доступ к проекту. Мы с начала и до конца работали только в одной библиотеке – pract_sample, поэтому никаких дополнительных действий от нас не требуется. Если проект содержит элементы из нескольких библиотек (не считая стандартных и проектных библиотек), то необходимо создать новую библиотеку, в которую скопировать все эти элементы. Делается это автоматически: выбирается ячейка верхнего уровня и выполняется иерархическое копирование с обновлением элементов (пункт «Copy» контекстного меню окна

Library Manager, опции – «Copy Hierarchical» и «Update Instances – of New Copies Only»).

- Для облегчения доступа к проекту ячейку верхнего уровня принято помещать в категорию с именем top. Выделим библиотеку parct_sample и выберем пункт меню «Edit/Categories/New…» окна Library Manager. В открывшемся окне New Category в поле «Category Name» напишем top. Затем с помощью кнопки «→» перенесем из списка «Not In Category» в список «In Category» ячейку myadc. Закроем окно, нажав **OK**. Теперь, если опция «Show Categories» в верхней части окна Library Manager включена, то в нашей библиотеке появится новая категория top.
- Проверим свои действия. Закроем Cadence и запустим его из другой папки или от имени другого пользователя, чтобы список подключенных библиотек не содержал ничего лишнего. Подключим нашу библиотеку с помощью меню «Edit/Library Path…». Запустив ячейку myadc.config, убедимся, что все элементы иерархии доступны, ничто не пропало.

Займемся подготовкой к экспорту. Топология верхнего уровня в нашем проекте является иерархической. Поэтому экспортный файл также содержит иерархию, которая будет передана на фабрику. Это не всегда хорошо, хотя бы с точки зрения авторских прав, ведь иерархические проекты легко читаемы, и восстановить схему по ним не составляет труда. Топологический редактор позволяет удалить иерархию, т.е. сделать все элементы объектами одного уровня. Но мы знаем, что иерархия – это существенная экономия памяти, так как повторяющиеся элементы хранятся на диске один раз. Топологический файл законченной микросхемы с полностью удаленной иерархией представляет собой просто рисунок, т.е. набор многоугольников, для каждого из которых необходимо хранить в памяти координаты, размеры и номер слоя. Оказывается, что значительную долю объема данных в этом случае занимают контактные площадки – они содержат миллионы контактных окон, параметры которых также необходимо хранить. При этом никакой смысловой нагрузки они не несут, и бояться нарушений авторских прав не приходится. В итоге в качестве компромисса часто останавливаются на варианте, когда из топологии удаляется вся иерархия, кроме контактных площадок. Это позволяет существенно уменьшить объем файла. Так мы и поступим.

- Удалим всю иерархию, кроме контактной площадки. Обычно стандартная ячейка контактной площадки имеет свою собственную иерархию, на одном из уровней которой есть элемент, представляющий собой ее топологию (в библиотеке XFAB он называется io_pad). Скопируем вид топологии myadc.layout в myadc.layout_flat. Начинаем удалять иерархию, до тех пор, пока среди дочерних элементов в структуре myadc.layout не останется только io_pad. При этом схема (вид schematic) остается иерархической. Для удаления иерархии выделяем всю топологию чипа и выбираем пункт «Edit/Hierarchy/Flatten…» окна Virtuoso. Затем повторяем процедуру (можно использовать опции **displayed levels** и **Flatten Pcells** окна Flatten) до тех пор, пока не останется только io_pad. Контроль текущей иерархии выполняется из меню «Design/Hierarchy/Tree…» окна Virtuoso.
- Производим экспорт в стандартный формат GDS-2. Выберем пункт «File/Export/Stream…» главного окна Cadence CIW. Откроется окно Virtuoso Stream Out. В соответствующих полях укажем ячейку для экспорта: pract_sample.mayadc.layout_flat. Указываем режим «Stream DB» и имя файла, в который будет произведен экспорт. Сжатие лучше не делать при экспорте, а заархивировать файл позже – это поможет лучше контролировать процесс. Масштаб **Scale UU/DBU** выставляется в соответствии с инструкцией производителя (у нас «0.001»). Важно имя файла с журналом ошибок – он может понадобиться, если экспорт не пройдет гладко. Запускаем процесс, нажав **OK**.
- По окончании процесса появится окно STRMOUT PopUp Message с сообщением о количестве ошибок и предупреждений. Нажав кнопку **Display Log**, можно просмотреть журнал сообщений.
- Для контроля сохранности данных при передаче по сети полезно использовать функцию контрольной суммы. В консоли Linux воспользуемся командой md5sum <имя файла> и запишем результат. Заархивируем файл программой gzip и снова запишем контрольную сумму.
- Сделаем контрольную проверку. Разархивируем файл и проверим контрольную сумму. В Cadence создадим новую чистую библиотеку pract_sample_test, в которую произведем импорт gds-

файла (команда «File/Import/Stream…» окна CIW). В библиотеке должны появиться две ячейки с видами layout: myadc и io_pad. Теперь проведем DRC и LVS верификацию импортированного файла, не забыв в качестве схемы для сравнения при LVS указать pract_sample.myadc.schematic.

- В случае успеха файл можно отправлять на фабрику, указав в примечании контрольную сумму до и после архивации.

Заключение

В данном учебном пособии были изложены базовые принципы проектирования СнК с помощью современных САПР. В первой и второй части пособия рассмотрены общие вопросы создания СнК и показана методология высокоуровневого проектирования на основе готовых ячеек и СФ блоков. В третьей части дан практический пример проектирования одного из таких блоков. Последовательно, шаг за шагом рассмотрены этапы разработки несложной аналого-цифровой схемы с использованием средств автоматизированного проектирования компании Cadence. Охватить все невозможно, но полагаем, что удалось дать представление о маршруте создания СнК и обратить внимание на проблемы, встающие перед разработчиками интегральных микросхем на этом трудном, но интересном пути. Хотелось бы надеяться, что книга окажется полезной будущим специалистам в области микро- и наноэлектроники.

Список литературы

1. Микросхемы интегральные. Сверхбольшие интегральные схемы типа «система на кристалле» и сложно-функциональные блоки. Термины и определения. – М.: НИИМА «Прогресс», 2007.

2. Немудров, Мартин Г. Системы-на-кристалле. Проектирование и развитие. – М.: Техносфера, 2004.

3. Федотов Я., Щука А. Система на кристалле // Электронные компоненты. 2001. № 2. С. 3–5.

4. Кривченко И. Системы на кристалле: общее представление и тенденции развития // Компоненты и технологии. 2001. № 6. С. 48–52.

5. Бухтеев А. Методы и средства проектирования систем на кристалле // Chip News. 2003. № 4. С. 4–8, 11–14.

6. Корнеев И., Немудров В., Польщиков В., Лагутин О. Специализированная СБИС типа «система на кристалле» навигационного приемника Глонасс/GPS // Электронные компоненты. 2007. № 4. С. 81–85.

7. Шевченко П., Шкуренко А. Декодер цифрового телевизионного сигнала высокой четкости: система на кристалле // Электроника НТБ. 2007. № 8. С. 62–66.

8. Kundert K., Chang H. Top-Down Design and Verification of Mixed-Signal Circuits / Designer's Guide Consulting, 2007.

9. Эннс В.И., Кобзев Ю.М. Проектирование аналоговых КМОП-микросхем. Краткий справочник разработчика / Под ред. В.И. Эннса. – М.: Горячая линия – Телеком, 2005. С. 188 – 209.

10. FitzPatrick D., Miller I. Analog behavioral modeling with the Verilog-A language. – Kluwer Academic Publishers, 2003.

11. Cadence Verilog-A Language Reference 6.0 / Cadence Design Systems, 2005.

12. Cadence Library Manager User Guide 5.0 / Cadence Design Systems, 2004.

13. Гуменюк А., Бочаров Ю. Методика анализа Фурье при моделировании аналого-цифровых схем с помощью средств проектирования Cadence // Chip News. 2007. № 9. С. 22 – 25.

14. Лайонс Р. Цифровая обработка сигналов. – М.: Бином, 2006.

15. Virtuoso Schematic Editor User Guide 5.1.41USR3 / Cadence Design Systems, 2005.

16. Cadence Hierarchy Editor User Guide 5.1.41 / Cadence Design Systems, 2005.

17. Virtuoso Analog Design Environment User Guide 5.1.41 / Cadence Design Systems, 2005.

18. WaveScan Tutorial 5.1.41 / Cadence Design Systems, 2005.

19. Waveform Calculator User Guide 5.1.41 / Cadence Design Systems, 2005.

20. Cho T., Gray P. A 10 b, 20 Msample/s, 35 mW Pipeline A/D Converter // IEEE J. Solid-State Circuits. 1995. Vol. 30. № 3. P. 166 – 172.

21. Virtuoso Mixed-Signal Circuit Design Environment User Guide 5.1.41 / Cadence Design Systems, 2005.

22. Saint Ch., Saint J. IC Layout Basics. A Practical Guide. – McGraw-Hill, 2002.

23. Hastings A. The Art of Analog Layout. – Prentice Hall, 2001.

24. Baker R. J. CMOS Circuit Design, Layout, and Simulation. Second Edition. – IEEE Press, 2005.

25. SimVision User Guide 5.6 / Cadence Design Systems, 2005.

26. Clein D. CMOS IC Layout Concepts, Methodologies, and Tools. – Newnes, 1999.

27. Assura Physical Verification User Guide 3.1.5 / Cadence Design Systems, 2006.

28. Virtuoso XL Layout Editor User Guide 5.1.41 / Cadence Design Systems, 2005.

29. 0.18 um Process Latch-up Design Guideline. Ver. 2.0 / UMC, 2005.

30. Badaroglu M., et al. Digital Circuit Capacitance and Switching Analysis for Ground Bounce in ICs with a High-Ohmic Substarte // IEEE J. Solid-State Circuits. 2004. Vol. 39. № 7. P. 1119 – 1130.

31. Van Heijninhen, et al. Analysis and experimental verification of digital substrate noisegeneration for epi-type substrates // IEEE J. Solid-State Circuits. 2000. Vol. 35. № 7. P. 1002 – 1008.

32. Badaroglu V., et al. Modeling and experimental verification of substrate noise generation in a 220-Kgates WLAN system-on-chip with multiple supplies // IEEE J. Solid-State Circuits. 2003. Vol. 38. № 7. P. 1250 – 1260.

33. Semenov O., et al. ESD Protection Device and Circuit Design for Advanced CMOS Technologies. – Springer Science, 2008.

34. Wang A. Z. H. On-Chip ESD Protection for Integrated Circuits. – Kluwer Academic Publishers, 2002.

35. Kahng A. B., et al. Enhanced Design Flow and Optimization for Multi-Project Wafers // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems. 2006. Vol. 26. № 2. P. 301 – 311.

36. Hansford W. Cut your design time with multiproject wafer services // EE Times Asia. 01-Jun-2007.

Содержание

Ю.И. Бочаров, А.С. Гуменюк, А.Б. Симаков, П.А. Шевченко

ПРОЕКТИРОВАНИЕ БИС КЛАССА «СИСТЕМА НА КРИСТАЛЛЕ»

Учебное пособие

Редактор М.В. Макарова

Подписано в печать 21.11.2008. Формат 60x84 1/16.
Уч.-изд. л. 12,0. Печ. л. 11,75. Тираж 150 экз.
Изд. № 4/14. Заказ №

Московский инженерно-физический институт (государственный университет)
115409, Москва, Каширское ш., 31

Типография издательства «Тровант».
г. Троицк Московской обл.

Для заметок